Titre: Title:	Parallélisme sur plateforme reconfigurable de calcul à mémoire répartie
Auteur: Author:	Benoît Morin
Date:	2006
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Morin, B. (2006). Parallélisme sur plateforme reconfigurable de calcul à mémoire répartie [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7729/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7729/
Directeurs de recherche: Advisors:	
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

PARALLÉLISME SUR PLATEFORME RECONFIGURABLE DE CALCUL À MÉMOIRE RÉPARTIE

BENOÎT MORIN
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AVRIL 2006

Your file Votre référence
ISBN: 978-0-494-17960-4
Our file Notre référence
ISBN: 978-0-494-17960-4

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

PARALLÉLISME SUR PLATEFORME RECONFIGURABLE DE CALCUL À MÉMOIRE RÉPARTIE

présenté par: MORIN Benoît
en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de:

M. BERTRAND François, Ph.D., président
M. ROY Robert, Ph.D., membre et directeur de recherche
M. BOIS Guy, Ph.D., membre et codirecteur de recherche
Mme NICOLESCU Gabriela, Doct., membre

da Sophie, trugarez

REMERCIEMENTS

C'est parfois face à l'adversité que l'on redécouvre tous ceux qui nous entourent et nous soutiennent. À tous ceux qui m'ont soutenu hier, me soutiennent aujourd'hui et me soutiendront demain, je l'espère, je vous dis merci, c'est pour vous tous que je soutiens.

Robert Roy, pour avoir cru en moi, pour m'avoir épaulé, pour m'avoir conseillé, pour m'avoir aidé à grandir et surtout pour avoir été patient, je vous dis merci.

Guy Bois, pour m'avoir fait confiance, pour m'avoir laissé libre et également pour votre patience, je vous dis merci.

Membres du jury, pour votre temps, vos commentaires ainsi que votre clémence, je vous dis merci.

Sylvain Fourmanoit, Marc St-Pierre, Jean-François Richard et Olivier Crête, pour votre vision, votre labeur et surtout votre enthousiasme qui ont contribué à faire de ce projet ce qu'il est aujourd'hui, je vous dis merci.

Louis-Alexandre Leclaire, Luc Lalonde et Kostas Piyakis, pour vos remarques et vos suggestions judicieuses, qui m'ont poussé sans cesse à rechercher la qualité, je vous dis merci.

Mille fois merci.

RÉSUMÉ

L'évolution des supercalculateurs a vu naître et disparaître bon nombre d'architectures différentes. Des processeurs vectoriels en passant par les grappes de calcul, tous ces changements se sont effectués dans le but avoué d'accroître la performance.

Les systèmes embarqués ont connu eux aussi une évolution similaire. Aujourd'hui, les puces de logique reprogrammables sont à la fine pointe de la technologie. Grâce à ces dernières, il est possible de prototyper des systèmes avant de les réaliser en circuit intégrés.

Il est également possible d'utiliser ces puces comme canevas matériel. Ainsi, celles-ci peuvent être reprogrammées pour implémenter quelque circuit numérique que ce soit. Ces circuits peuvent tirer avantage de par leur nature d'un parallélisme inhérent.

Ces circuits logiques sont maintenant couplés à des processeurs intégrés à même la puce. Il devient alors intéressant, à l'exemple des grappes de calcul, d'étendre la puissance de ces systèmes en reliant entre eux plusieurs puces de ce genre.

Le projet consiste à intégrer un certain nombre de cartes de développement comportant des puces de logique reprogrammable de type Virtex II Pro au sein d'une grappe de calcul traditionnelle.

La méthode préconisée est d'appliquer des paradigmes jusqu'ici réservés aux systèmes à mémoire distribuée à ces grappes de puces reprogrammables. Ces plate-

formes embarquées sont alors considérées au même titre qu'un noeud classique d'une grappe de calcul.

Voici les objectifs de ce projet :

- Intégrer les cartes de développement à base de FPGA au sein d'une grappe de calcul.
- Évaluer la carte de développement relativement à son utilisation sous forme de grille.
- Ajouter des fonctionnalités pour supporter les grappes hétérogènes sous Adelie Linux.
- Comparer la performance de modules d'accéleration matériels à des implémentations logicielles.

La trousse de logiciels Adelie/SSI a été réécrite afin de permettre l'intégration de matériel de type hétérogène. Cette trousse nécessite l'utilisation de la distribution Gentoo Linux.

Plusieurs années d'expérience relative au déploiement et à la gestion de grappes de calcul ont été condensées dans un ensemble de modules de configuration. Ces modules permettent de regénérer l'ensemble de la configuration d'une grappe de calcul à partir d'un unique fichier de configuration simple. Intégrés à la distribution Gentoo, ces modules permettent une mise à niveau automatisée d'une grappe de calcul comme s'il s'agissait d'un simple poste de travail.

Afin d'améliorer la performance de ces derniers, un mécanisme de cache a été

incorporé afin de maintenir le temps d'exécution à un minimum. De plus, les temps de démarrage des noeuds selon le mode classique ainsi qu'un nouveau mode utilisant unionfs ont été évalués. Le mode basé sur unionfs brille tout particulièrement lorsque le nombre de fichiers à copier localement est important.

La bande passante au niveau du protocole TCP entre les différentes configurations de noeuds hôtes et de noeuds embarqués présente des résultats mixtes. Les performances de l'interface gigabit ethernet en réception sont très décevantes sur les noeuds embarqués. Le module gigabit ethernet utilisé occupe un grand nombre de cellules logiques sans toutefois offrir plus de performance qu'une interface fast ethernet.

L'intégration de puces de type FPGA au sein d'une grappe de calcul traditionnelle peut sous de bonnes conditions venir augmenter la puissance de calcul globale de celle-ci. L'intégration de ces puces à titre de périphériques simples n'est valable que pour un nombre limité. L'exploitation de ces dernières à titre de noeud à part entière d'une grappe se veut une solution qui peut être mise à l'échelle.

ABSTRACT

The evolution of supercomputers has seen the birth and the disappearence of a good number of different architectures. From vectorial processors to clusters, all these changes were made with the goal of increasing performance.

Embedded systems have also known a similar evolution. Today, programmable gate arrays are the state of the art of technology. Using these systems as design tools, it is possible to prototype systems before creating the integrated circuits.

It is also possible to use these chips as a form of hardware canvas. They can be reprogrammed to implement almost any digital circuit possible. These circuits can, by their nature, take advantage of inherent parallelism.

These digital circuits are now coupled to processors, which are integrated on the same die. It then becomes interesting to expend the power of these systems by connecting many of them toghether, just like a cluster.

This project consists in the integration of a number of development cards containing Virtex II Pro programmable gate arrays with a traditionnal cluster.

The proposed method is to apply paradigm upto now reserved to distributed memory systems to these clusters of gate arrays. These embedded plateforms are then considered on par with a typical node of a cluster.

Here are the objectives of this project :

- Integrate the FPGA-based development cards within the cluster.
- Evaluate the development card regarding its use within a grid topoloy.
- Add functionnalities to Adelie Linux to support heterogenous clusters.
- Compare performance of hardware acceleration modules with software implementations.

The Adelie/SSI toolkit was rewritten to allow for integration of heterogenous hardware. This toolkit requires the use of the Gentoo Linux distribution.

Many years of experience concerning deployment and management of clusters were condensed within a set of configuration modules. These modules allow the generation of the entire configuration of a cluster from a simple unique configuration file. Moreover, updating the cluster is now as easy as updating a single personnal computer due to the Gentoo management system.

In order to increase the performance of these modules, a cache mecanism was incorporated in order to keep the execution time to a minimum. The startup times of the nodes in both the classic mode and the new mode using unionfs where evaluated. The mode based on unionfs shines particularly when a large number of files must be locally copied.

The TCP layer bandwidth between different configurations of both host nodes and embedded nodes show mixed results. The performances of the gigabit ethernet interface when receiving are very disappointing on the embedded nodes. The gigabit ethernet module used occupies a large number of logic cells without offering any more performance than a fast ethernet interface.

However, the integration of FPGA type chips within a traditional cluster can, under good conditions, increase the global computing power of the cluster. The integration of theses chips as simple peripherals is only valid for a limited number of them. The use of these chips in a full fledged cluster nodes is a more scalable approach.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES LISTAGES

LISTE DES SIGLES ET ABRÉVIATIONS

ARP: Address Resolution Protocol

BIOS: Basic Input and Output System

BOOTP: Bootstrap Protocol

CERCA: Centre de Recherche en Calcul Appliqué

CHP: Calcul de Haute Performance

DHCP: Dynamic Host Configuration Protocol

FPGA: Field Programmable Gate Array

GNU: Global File System

GNU: Gnu is Not Unix

HPC: High Performance Computation

IDE: Integrated Drive Electronics

IP: Internet Protocol

JTAG: Joint Test Action Group

LDAP: Lightweight Directory Access Protocol

MAC: Media Access Control

MTU: Maximum Transmission Unit

NFS: Network File System

NIS: Network Information Service

NTP: Network Time Protocol

PBS: Portable Batch System

PCI: Peripheral Component Interconnect

PLB: Processor Local Bus

PXE: Preboot Execution Envrionment

RAID: Redudant Array of Independent Disks

RFC: Request For Clarification

RPC:	Remote Procedure Call
RRQ:	Read Request
RSH:	Remote Shell
SAN:	Storage Area Network
SAR:	System Administration Resources
SATA:	Serial Advanced Technology Attachment
SSH:	Secure Shell
SSI:	Single System Image
TCP:	Transmission Control Protocol
TFTP:	Trivial File Transfer Protocol
USB:	Universal Serial Bus
UTC:	Universal Time Coordinated
WOL:	Wake on LAN

LISTE DES ANNEXES

INTRODUCTION

L'évolution de l'informatique se veut une course à la performance. Si la nature a horreur du vide, les informaticiens ont quant à eux horreur des cycles perdus. Il ne faut donc pas s'étonner de voir des systèmes de plus en plus puissants ainsi que des applications de plus en plus complexes pour harnacher cette puissance.

À l'origine, les supercalculateurs n'étaient composés que d'un seul processeur, ne traitant qu'un seul flot de données. Afin d'augmenter la capacité de traitement, les premiers systèmes parallèles furent développés. Les ordinateurs vectoriels étaient composés de plusieurs unités arithmétiques distinctes permettant d'effectuer une même opération sur un ensemble de données. Il était ainsi possible d'obtenir un parallélisme de données. Un processeur maître s'occupait d'amener le flot de données aux unités vectorielles; toutefois, cette démarche, très intéressante pour les boucles de calcul, s'avère inutile dans le cas de nombreux branchements.

Une branche connexe de l'évolution de l'informatique a vu apparaître des systèmes composés de plusieurs processeurs complets, partageant une même mémoire. Ces ordinateurs multi-processeurs pouvaient effectuer un traitement différent sur des données différentes. En plus de permettre un parallélisme de données comme les systèmes vectoriels, les systèmes parallèles permettent un parallélisme de tâche.

Avec l'augmentation du nombre de processeurs dans les supercalculateurs, la contention des ressources devint un problème des plus importants. Afin de réduire ces goulots d'étranglement, la mémoire fut divisée entre chaque processeur. Exit les systèmes vectoriels au profit des architectures à accès mémoire non-uniformes. Les échanges entre processeurs devaient se faire de manière explicite par le biais de messages.

Étant reliés par un bus de communication secondaire, ces systèmes n'étaient plus soumis aux mêmes contraintes. En 1993, Donald Becker et Thomas Sterling eurent l'idée de reproduire une architecture plus faiblement couplée en utilisant du matériel informatique générique. En reliant entre eux plusieurs ordinateurs individuels, on pouvait obtenir un schéma similaire à un supercalculateur parallèle. C'est ainsi qu'on vit apparaître les grappes de calcul de type Beowulf (STERLING et al., 1995).

L'informatique embarquée a suivi elle aussi une évolution similaire. À l'origine composés de systèmes électroniques simples, les systèmes embarqués ont vite évolué vers des systèmes informatiques complexes. Les besoins de miniaturisation ont poussé à l'intégration des composantes.

Les puces de logique reprogrammables s'inscrivent dans cette optique. Il s'agit de circuits intégrés numériques pouvant être reprogrammés afin d'exécuter des tâches précises. Ces puces sont aujourd'hui utilisées pour prototyper divers systèmes embarqués avant la production de circuits intégrés spécifiques. Il est en effet possible de simuler le comportement d'un circuit intégré numérique durant son développement. De plus, certains systèmes embarqués fabriqués en faible volume peuvent bénéficier d'une intégration jadis réservée aux grands volumes de production.

Il est possible de réaliser à l'aide de ces puces de logique des circuits reproduisant un algorithme logiciel particulier. De cette manière, le parallélisme inhérent à ce type de circuits peut être exploité. Un processeur générique peut donc utiliser une puce de logique afin d'optimiser certaines opérations qui sont séquentiellement coûteuses.

Il est également possible d'instancier directement un processeur à même la logique disponible sur ces puces. La firme Xilinx y est allé d'un pas de plus en incluant un

ou plusieurs porcesseurs PowerPC dans sa famille de puces reprogramables Virtex II Pro. Ainsi aucune cellule logique n'est perdue pour instancier ce ou ces processeurs.

À l'instar des supercalculateurs, ces puces de logiques reprogrammables ont une limite de performance. Il est donc intéressant de leur appliquer les mêmes principes qu'aux systèmes parallèles. En interreliant plusieurs de ces puces entre elles, il est donc théoriquement possible d'augmenter leur capacité totale de traitement.

En utilisant un matériel plus générique, il devient également possible de créer des grappes de logique reconfigurables, au même sens qu'une grappe de calcul de type Beowulf. En alliant ce type de système à une grappe de calcul traditionnel, il est alors possible d'obtenir un système hétérogène dédié au traitement de données.

L'objectif du projet consiste donc à adapter des paradigmes de développement du domaine des systèmes à mémoire distribuée, afin de satisfaire aux contraintes particulières d'une plateforme matérielle reconfigurable. Il s'agit donc d'exploiter de manière efficace sous forme de grappe plusieurs puces de type FPGA, tout en respectant des contraintes temps réel.

L'intégration à très grande échelle permet de nos jours d'obtenir des puces reprogrammables de type FPGA contenant un nombre élevé de portes logiques. Malgré ceci, il arrive que les besoins de performance dépassent les capacités des meilleures puces disponibles sur le marché. La mise en réseaux de puces de ce type permet d'en augmenter la puissance de traitement équivalente.

Afin de faciliter l'intégration de systèmes de ce genre, des outils de parallélisation typiquement utilisés dans les systèmes informatiques à mémoire distribuée devront être adaptés.

Il est prévu d'étendre les outils de grappes déjà disponibles pour les processeurs

génériques via la distribution Adelie Linux sur des puces programmables de type FPGA. En voici les étapes :

- Intégrer les cartes de développement à base de FPGA au sein d'une grappe de calcul.
- Évaluer la carte de développement relativement à son utilisation sous forme de grille.
- Ajouter des fonctionnalités pour supporter les grappes hétérogènes sous Adelie Linux.
- Comparer la performance de modules d'accélération matériels à des implémentations logicielles.

Le premier chapitre présente l'architecture matérielle qui compose la grappe de calcul hétérogène. De plus, l'intégration des différentes plateformes ainsi que la topologie utilisée y sont décritent.

Le second chapitre présente le niveau fondamental de la pile logicielle de la grappe de calcul. Il s'agit d'une présentation des systèmes d'exploitations utilisés par les différentes plateformes.

Le troisième chapitre présente les outils d'intégration et de gestion automatisée de la grappe de calcul. L'exploration d'Adelie/SSI y est faite en détails.

Le dernier chapitre présente les résultats des évaluations de performance de la grappe de calcul. Les résultats reliés au matériel ainsi qu'au logiciel y sont détaillés.

CHAPITRE 1

ÉTAT DE L'ART

Depuis les débuts de l'informatique, la puissance de calcul a toujours été le facteur limitant. Les grappes de calcul sont apparues comme une solution intéressante et peu coûteuse à ce problème. L'intégration de matériel dit reconfigurable au sein d'une grappe de calcul traditionnelle peut être perçue comme une étape supplémentaire dans cette quête de performance. Malgré la relative jeunesse de ce domaine de recherche, il existe plusieurs efforts notables.

1.1 Solutions logicielles

Les solutions logicielles pour grappes de calcul se divisent en deux catégories, selon si le système offre une vue distribuée des processus ou encore une vue centralisée de ceux-ci.

De plus, il est possible de déployer ces grappes par réseau, sans utiliser de disques locaux, qu'elles soient à image unique ou non.

1.1.1 Distributions pour grappes de calcul

Le concept Beowulf (STERLING et al., 1995) a beaucoup évolué depuis son apparition au milieu des années 90.

Il existe plusieurs livres (STERLING et al., 1999) (BOOKMAN, 2002) (VRENIOS, 2002) (SLOAN, 2004) de recettes pour aider au déploiement de grappes de calcul.

La distribution ROCKS (PAPADOPOULOS et al., 2001) est une distribution Linux complète axée sur les grappes de calcul. Il s'agit en somme d'une recompilation de Red Hat Enterprise Linux, afin de respecter la licence de cette dernière.

OSCAR (DE LIGNERIS et al., 2003) est un méta-distribution pour grappes de calcul. Cette dernière s'installe au dessus d'une distribution Linux. Elle peut être utilisée avec Red Hat Enterprise, Fedora ou encore CentOS.

1.1.2 Systèmes à image unique

Une des plus anciennes distributions pour systèmes à image unique est Scyld Beowulf (SCYLD SOFTWARE, 2004). Cette distribution, à l'image de ROCKS, est ditribuée avec une version recompilée de Red Hat Enterprise.

Les solutions Mosix (BARAK and LA'ADAN, 1998) et OpenMosix (BAR and KNOX, 2004) apportent des modifications au noyau Linux standard afin de permettre l'utilisation d'un espace de processus unique. Un ensemble de logiciels les accompagne pour gérer la distribution des processus sur l'ensemble des noeuds.

Le projet OpenSSI (WALKER, 2003) offre une solution similaire à OpenMosix, c'est-à-dire un noyau Linux modifié, mais intègre en plus un certain nombre d'outils de gestion pour les grappes de calcul.

1.1.3 Grappes de calcul sans disques

Le projet d'une grappe de calcul sans disque à été développé au CERCA en 2001.

Il existe une adaptation d'OSCAR qui permet le déploiement de grappes de calcul en mode sans disque. Il s'agit du projet Thin-OSCAR (DE LIGNERIS and

GIRALDEAU, 2003).

Une autre solution encore plus récente, Warewulf (WAREWULF CLUSTERING SOLUTION, 2005), permet de déployer un bon nombre de distributions en mode sans disque. Pour ce faire, une image de système minimale est créée à partir de l'installation choisie puis exécutée en mémoire sur les noeuds.

1.2 Solutions matérielles

L'utilisation de matériel spécialisé, en particulier de puces reconfigurables de type FPGA apparaît comme une solution viable pour accroître les performances de calcul des stations de travail. Il existe présentement sur le marché quelques solutions de ce type.

1.2.1 Coprocesseurs reconfigurables

Le système XD1 (CRAY, 2004), développé par Synective et commercialisé par Cray, intègre une puce Virtex 4 de Xilinx au sein d'un noeud dual Opteron.

Le module de corpocesseur offert par DRC (DRC COMPUTER CORPORATION, 2006), s'intègre dans un système multi-processeurs Opteron. Il remplace un des processeur et utilise le bus Hypertransport pour communiquer et pour accéder la mémoire présente sur les autres processeurs.

1.2.2 Grappes de puces reconfigurables

Le module CNP 2.0 (SYNECTIVE LABS, 2004) de Synective est une carte à base de Virtex 4 qui peut être utilisée sans processeur externe. Chaque carte comporte

4 FPGA et il est possible de relier seize de ces cartes afin de créer un grappe de 64 FPGA.

1.3 Intégration matérielle/logicielle

Le projet SInRG (LEHRTER et al., 2002) regroupe au sein d'une même grappe de calcul des noeuds traditionnels ainsi que des noeuds comprenant du matériel reconfigurable. Ce matériel est exploité à l'aide d'un mécanisme de librairies couplées logicielles/matérielles.

CHAPITRE 2

MATÉRIEL

Le projet d'intégrer une grille de puces de logique reconfigurables au sein d'une grappe de calcul a connu différentes révisions au cours de son élaboration. Tout au long du développement de la carte embarquée par la firme Amirix, l'architecture de celle-ci a été modifiée. Par la suite, le projet de grille de 3x3 a été révisé pour adopter une grille de 2x2.

L'objectif est resté le même toutefois, c'est-à-dire intégrer au sein d'une grappe de calcul traditionnelle des cartes de développement basées sur une puce de logique reconfigurable Virtex II Pro de Xilinx. Cette puce comporte un certain nombre de processeurs de type PowerPC 405 d'IBM devant être incorporés au sein de la grappe de calcul pour former une grappe hétérogène mais intégrée.

Il y a donc deux vues distinctes de la grappe hétérogène, la couche hôte, qui reçoit les cartes d'Amirix et la couche embarquée, constituée des plateformes implémentées sur la puce de logique reconfigurable.

2.1 Couche hôte

La couche hôte représente une grappe de calcul de type Beowulf quasi-classique. Elle était à l'origine composée de neuf noeuds mais a depuis été revue à cinq. Cette couche a pour fonction d'accueillir les cartes AP120 d'Amirix. Elle peut également servir de noeuds de calcul.

À l'origine, cette grappe était constituée de neuf noeuds répartis entre deux architectures différentes. Deux des neuf noeuds étaient composés de processeurs d'Intel et correspondaient à l'architecture suivante :

- Processeur Intel Xeon (2.4 GHz)
- Mémoire vive de 1 Go
- Disque dur de 80 Go
- Interconnexion fast ethernet (100 Mb/s) et gigabit ethernet (1 Gb/s)

Ces noeuds devaient occuper le double rôle de serveur et de noeud de calcul. C'est pour cela qu'ils ont été choisis un peu plus puissants que les sept autres noeuds. Les interconnexions gigabit ethernet devaient être utilisées pour relier la grappe au monde extérieur tandis que les interconnexions fast ethernet étaient réservées pour les communications au sein même de la grappe.

Les sept autres noeuds étaient quant à eux composés de processeurs d'AMD avec les caractéristiques suivantes :

- Processeur AMD Athlon XP 2500+ (1.8 GHz)
- Mémoire vive de 512 Mo

- Aucun disque dur
- Interconnexion fast ethernet (100 Mb/s)

Le choix d'avoir une architecture mixte est un choix volontaire. Il aurait très bien été possible d'utiliser des processeurs Athlon plus performants sur les noeuds frontaux également. Ce choix a été fait dans le but d'encourager le développement de solution de grappe de calcul avec un support hétérogène.

En effet, il n'avait pas été donné avant cette grappe de travailler à intégrer du matériel différent au sein d'une même grappe à image unique. Cela posait un défi qui s'intégrait dans la vision à plus long terme de développer une solution globale pour les systèmes totalement hétérogènes.

Cette grappe a été réduite à quatre noeuds plus un serveur dédié lorsque le projet a vu la taille de la grille passer de 3x3 à 2x2.

Initialement, cette grille devait être reliée par un commutateur fast ethernet. Suite à des problèmes de compatibilité, celle-ci a dû être remplacée par un commutateur gigabit ethernet. La figure 2.1 présente la topologie du réseau.

Cette grappe accueille la trousse d'outils de grappe de calcul Adelie/SSI, basée sur Gentoo Linux. Elle fait également office de point d'accès pour la couche embarquée. De plus, il est possible de l'utiliser pour développer de nouveaux progiciels pour modifier ou remplacer la plateforme embarquée.

En effet, la couche embarquée est reliée par port série à la couche hôte. Cela permet d'accéder à l'outil *ppcboot* même lorsque la plateforme en test n'est pas fonctionnelle. Aussi, l'utilitaire *apcontrol* d'Amirix permet de contrôler certains aspects de la carte AP100. Il est ainsi possible de redémarrer la plateforme embarquée ou encore de forcer sa reconfiguration.

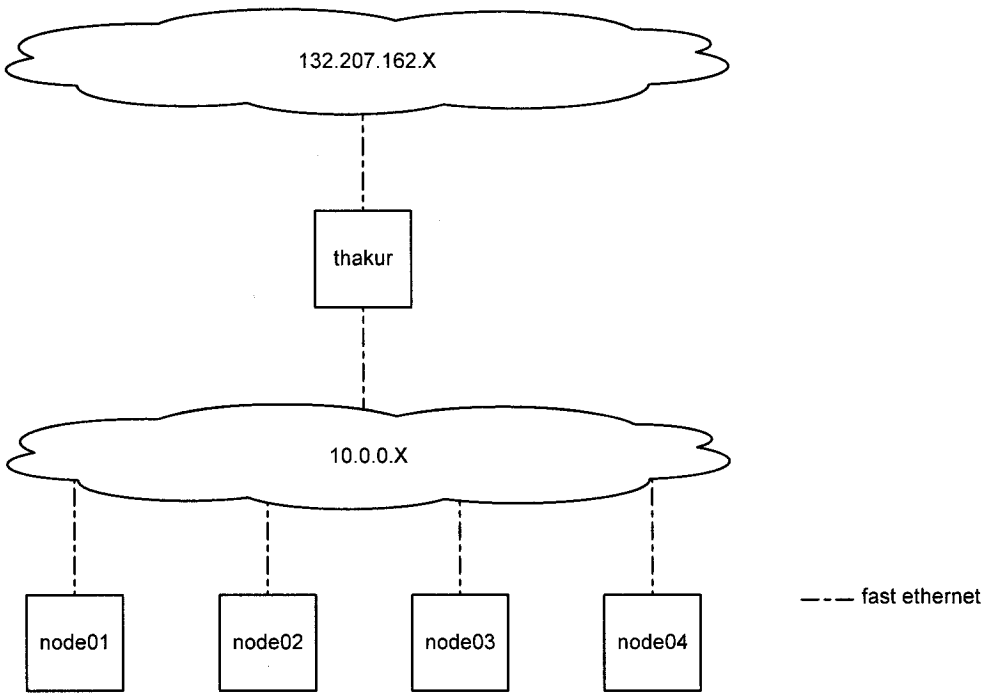

Figure 2.1 Diagramme réseau représentant le serveur et les noeuds hôtes de la grappe de calcul hétérogène.

Il existe sur le marché présentement des systèmes offrant une configuration similaire. La compagnie Cray offre le système XD1 qui intégre une puce Virtex II Pro à même le matériel du système.

La couche hôte joue donc de multiples rôles. Elle sert d'environnement hôte pour le dévelopement de la plateforme embarquée. Elle peut également être utilisée seule ou combinée avec la couche embarquée comme grappe de calcul.

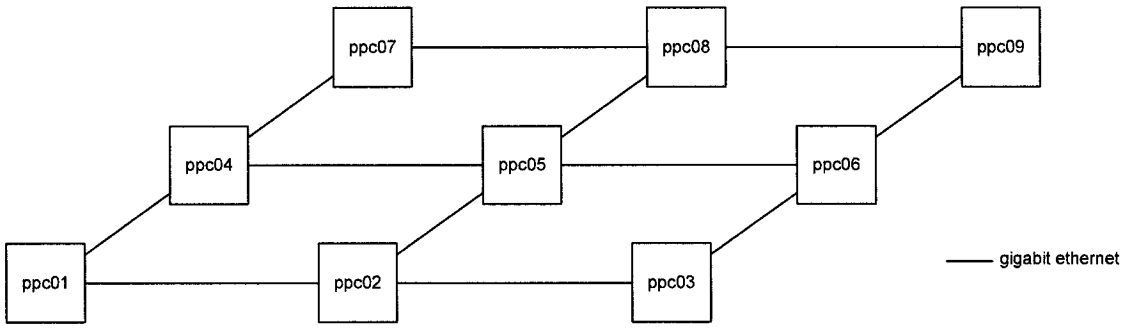

Figure 2.2 Diagramme réseau représentant les noeuds embarqués sous forme de grille de la grappe de calcul hétérogène.

2.2 Couche embarquée

La couche embarquée constitue également une architecture parallèle. Initialement, celle-ci devait être agencée sous forme de grille non-périodique de taille 3x3. La figure 2.2 présente la topologie initiale prévue. Suite à des contraintes matérielles, cette configuration a dû être réévaluée. L'architecture actuelle comporte deux grilles de 2x2, qui peuvent aussi être exploitées en mode pleinement connecté à un commutateur.

2.2.1 Carte AP120-6PCI

La couche embarquée est composée de quatre cartes AP120-6PCI développées par Amirix. Il s'agit de plateformes de développement complètes pour les puces de logique reprogrammable de type Virtex II Pro de Xilinx. Voici les détails de celles-ci :

- Xilinx Virtex II Pro
- 64 Mo Mémoire vive DDR SDRAM

- 4 Mo Mémoire flash (logiciel)
- 4 Mo Mémoire flash (configuration)
- Interace CompactFlash Xilinx SystemACE
- Bus PCI système 32 bit à 66 MHz
- Bus PCI local 32 bit à 33 MHz
- 2 ports gigabit ethernet
- 1 port fast ethernet
- 2 ports série
- Fente d'extension PMC IEEE 1386.1

La carte mezzanine AX13 est utilisée pour offrir deux ports gigabit ethernet additionnels.

La carte de développement AP120 s'intègre à la couche hôte par le biais d'une fente d'expansion PCI. De plus, deux ports série relient la plateforme progicielle au noeud hôte. Ces ports peuvent être utilisés pour déverminer un système ne comprenant pas de module ethernet ou encore lorsque qu'aucun logiciel n'est installé.

Un port fast ethernet est aussi disponible. Ce port ne fait pas partie de la plateforme embarquée, il s'agit d'un circuit externe. Malheureusement, celui-ci n'est pas disponible lorsque la plateforme quad gigabit est utilisée.

Trois interrupteurs sont disponibles sur la carte. Le premier permet de réinitialiser la carte, le second de forcer la reconfiguration de la plateforme à partir de la configuration de secours et finalement le troisième est la disposition du développeur de la plateforme embarquée.

Les fonctionnalités de ces interrupteurs peuvent être émulées à l'aide de l'utilitaire *apcontrol* d'Amirix. Cet utilitaire permet de forcer la reconfiguration à partir d'une des deux images présente en mémoire flash. Il permet également de réinitialiser la carte.

Un problème avec la version des cartes présentes dans la grappe de calcul hétérogène Thakur fait en sorte que la configuration doit être chargée à l'aide d'*apcontrol* suite à un redémarrage du noeud hôte. Un script d'initialisation nommé apcontrol a été créé sur ces noeuds afin que le chargement s'effectue de manière automatique lors de l'initialisation du système. De plus, un défaut sur la carte fait en sorte qu'il est impossible d'éteindre de manière logicielle un système qui interprète le signal WOL sur le bus PCI.

Pour modifier la configuration progicielle utilisée, trois mécanismes sont possibles. Les cavaliers JP11 et JP12 permettent de sélectionner la méthode à utiliser :

- Mémoire flash présente sur la carte.
- Carte de type compact flash.
- Cable de déverminage de type JTAG.

Une fente de type compact flash est disponible à l'arrière de la carte. Si l'on place l'image à être utilisée sur une carte de ce type et que l'on sélectionne ce mode de configuration, la puce sera reconfigurée en utilisant cette image. À l'avant de la carte, un port de déverminage de type JTAG est disponible. Il est possible de reconfigurer la puce à partir de celui-ci, ou encore de déverminer un système en fonctionnement.

2.2.2 Plateforme Quad Gigabit Ethernet

Deux plateformes progicielles développées par Amirix sont disponibles pour cartes de dévelopement AP120 composant la couche embarquée. Ces plateformes visent à exploiter les ressources présentes sur la puce ainsi que sur l'ensemble de la carte. Ainsi celles-ci intègrent les processeurs PowerPC 405 d'IBM qui sont présents sur la puce Virtex II Pro de Xilinx.

La première plateforme, nommée baseline ou plateforme de base, n'intègre qu'un seul des deux processeurs PowerPC présent sur la puce. Cette plateforme est celle qui se retrouve comme configuration de base lors de la réception de la carte. Elle intègre toute la mémoire ainsi que deux ports gigabit ethernet. De plus, sous cette configuration, le port fast ethernet est disponible. La figure 2.3 présente une vue schématique de cette plateforme.

Cette configuration offre certains avantages. De par sa taille réduite, elle offre un certain nombre de cellules logiques libres. Ces cellules peuvent être utilisées pour instancier des périphériques additionnels.

La plateforme qui est à la base du projet, nommé quad gigabit ethernet, a été développée par Amirix afin de réaliser une grille de cartes AP120. Chaque carte est reliée à ses quatre voisins immédiats par des liens gigabit ethernet. La figure 2.4 présente une vue schématisée de cette plateforme.

Dans cette plateforme, les deux processeurs PowerPC sont utilisés. Ils ne sont malheureusement pas reliés au même bus en mode SMP. Les PowerPC 405 ne possèdent pas la circuiterie nécessaire pour conserver une cohérence de cache. Les deux processeurs possèdent donc leur propre bus.

Les ressources y sont partagées entre les deux. Des 64 Mo disponibles pour

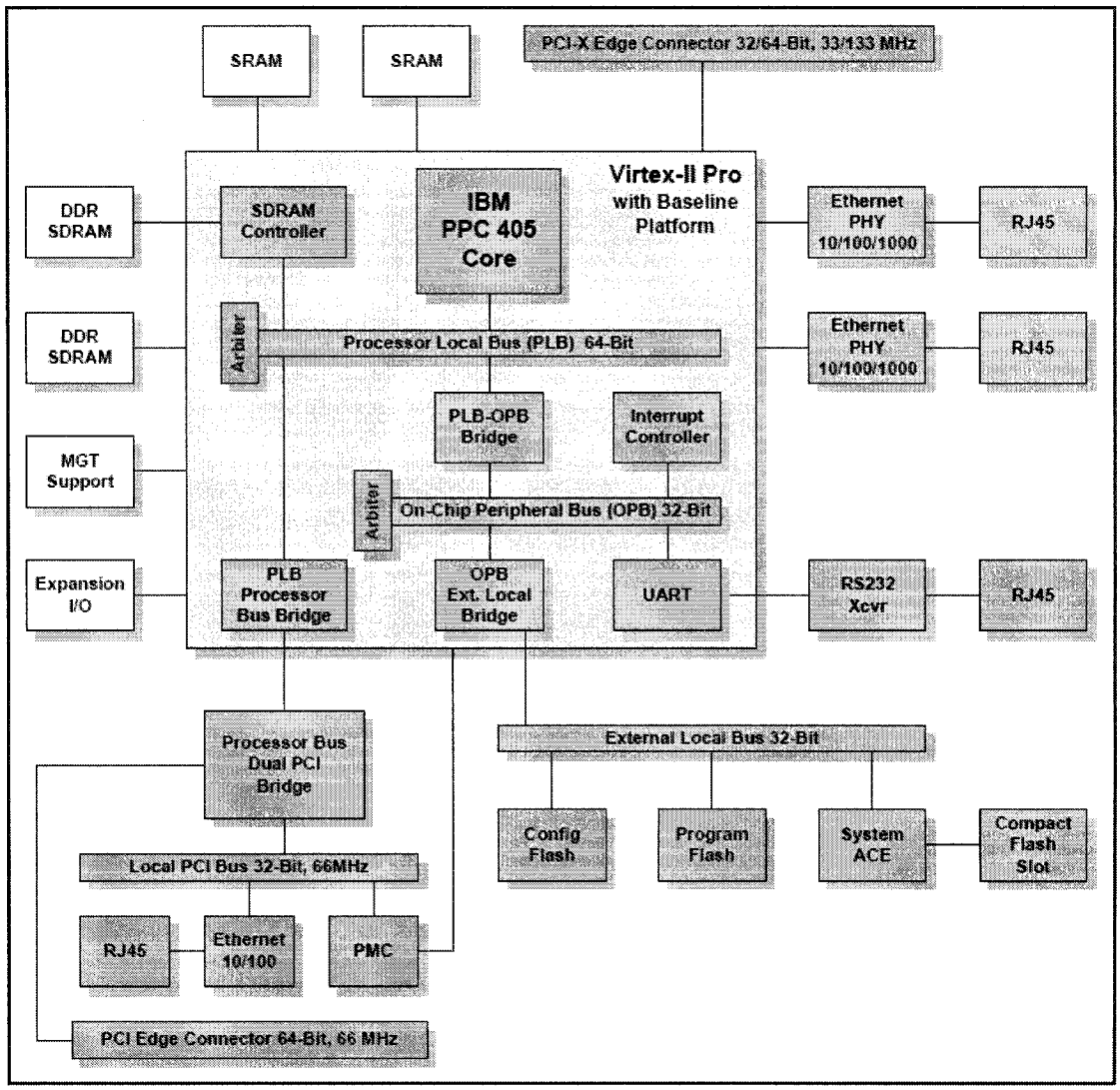

Figure 2.3 Plateforme matérielle reconfigurable de base comprenant deux interfaces Gigabit Ethernet.

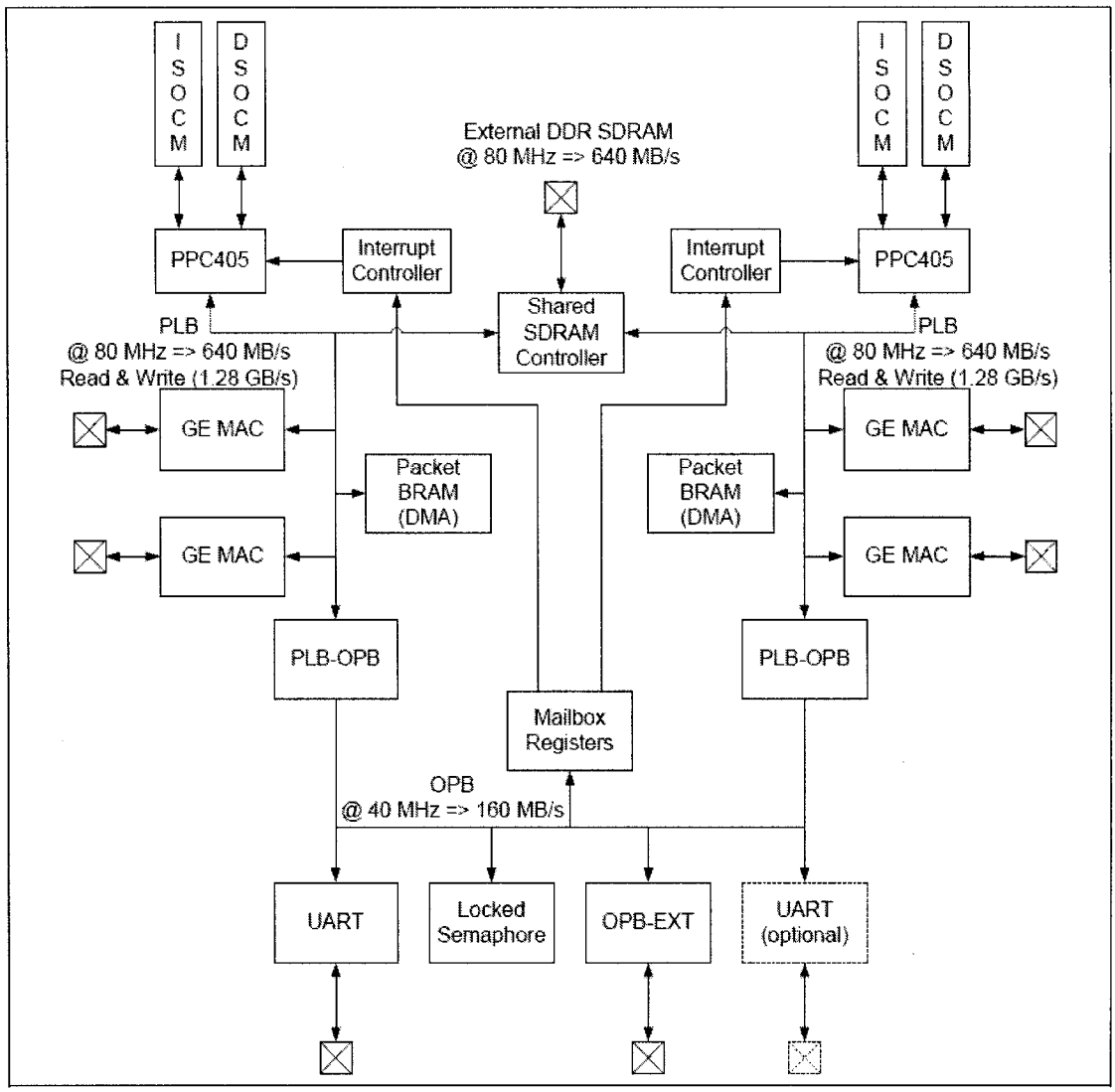

Figure 2.4 Plateforme matérielle reconfigurable comprenant quatre interfaces Gigabit Ethernet.

l'ensemble de la plateforme, seulement 32 Mo sont disponibles pour chaque processeur. De ce montant il faut également retrancher 1 Mo utilisé pour la zone de mémoire partagée entre les deux processeurs. On retrouve donc la configuration suivante sur chacune des plateformes :

- Processeur IBM PowerPC 405 (240 MHz)
- Mémoire vive de 31 Mo
- Interconnexion gigabit ethernet (1 Gb/s)

Les interfaces gigabit ethernet se retrouvent également partagées entre les deux systèmes. Chaque processeur à donc accès à deux interfaces gigabit. Pour relier les deux processeurs, un mécanisme utilisant les registres à boîte aux lettres est employé. Ce mécanisme émule une interface réseau, ce qui facilite son utilisation directe sans modification.

Afin d'obtenir l'image racine par NFS, une des deux interfaces est reliée directement au commutateur de la grappe. Ainsi, chaque noeud embarqué se trouve pleinement connecté au reste de la grappe.

L'interface gigabit ethernet développée par Amirix ne fonctionne qu'en mode gigabit, il est impossible de la relier directement à un lien fast ethernet. Le commutateur fast ethernet a dû être remplacé par un commutateur gigabit ethernet pour permettre la connexion vers les noeuds hôtes. En effet, ceux-ci utilisent des interfaces fast ethernet pour communiquer.

La figure 2.5 présente la topologie réseau de la couche embarquée.

Deux grilles de 2x2 sont donc reliées entre elles par un commutateur gigabit ethernet. L'utilisation d'un commutateur gigabit ethernet permet de relier les deux,

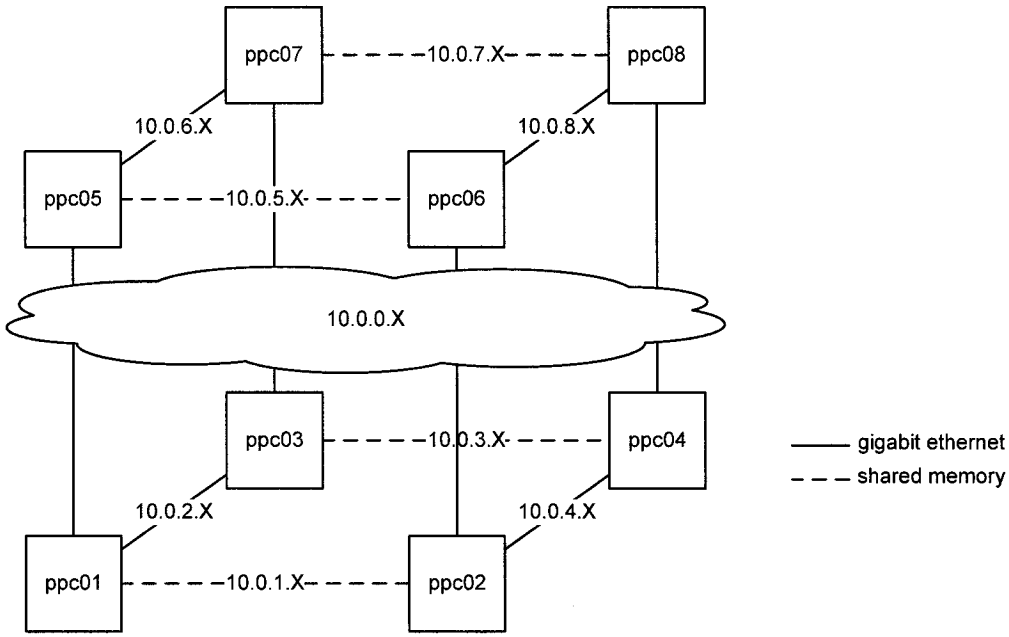

Figure 2.5 Diagramme réseau représentant les noeuds embarqués de la grappe de calcul hétérogène.

le commutateur s'occupant d'adapter les vitesses de transmission en utilsant des temps de réception.

L'utilisation de la plateforme quad gigabit ethernet pose plusieurs problèmes. Le nombre de cellules logiques utilisées pour instancier les modules gigabit ethernet est substantiel. La plateforme dans son entier occupe près de 95 pourcent des cellules disponibles, ce qui ne laisse aucune place pour l'intégration d'autres périphériques.

Le contrôleur de mémoire a dû être modifié pour permettre l'accès des deux plateformes à la mémoire unique. Ces modifications ont eu des répercussions importantes sur l'interface PCI entre la couche hôte et la couche embarquée. Il est dorénavant impossible d'écrire dans la mémoire de la carte embarquée directemement à partir du système hôte. Le seul mécanisme viable de communication entre la couche hôte et la couche embarquée reste via les interfaces ethernet.

Également, sous cette configuration, l'interface fast ethernet disponible sur les cartes AP120 n'est pas disponible.

Les deux processeurs embarqués qui se trouvent sur la plateforme n'ont pas été intégrés en mode SMP. Plutôt, deux plateformes distinctes possédant chacune leur processeur ont été créées. Il est donc maintenant impossible d'exploiter l'ensemble de la mémoire vive disponible, soit 64 Mo. Chaque processeur a accès à la moitié de celle-ci, soit 32 Mo.

Puisque les deux processeurs ne partagent plus la même mémoire, un mécanisme a été mis en place pour permettre la communication entre ceux-ci. Il s'agit d'une zone de mémoire partagée entre les deux. Cette zone de mémoire occupe 1 Mo sur chaque plateforme. Un mécanisme simulant un adaptateur TCP/IP a été intégré dans la plateforme pour permettre la communication entre les processeurs. Cet adaptateur utilise la zone de mémoire partagée pour échanger des paquets.

Malheureusement, les interfaces gigabit ethernet des noeuds embarqués ne fonctionnent qu'en gigabit ethernet. il est donc impossible de les relier à un commutateur fast ethernet. Pour faire fonctionner la communication, il faut utiliser un commutateur gigabit ethernet. Celui-ci sera en mesure d'adapter les taux de transmission en utilisant des tampons.

Ces facteurs viennent limiter le potentiel de cette plateforme. Les versions ultérieures des cartes, possédant plus de ressources en terme de cellules logiques et de blocs de mémoires, pourront s'avérer plus intéressantes dans le contexte d'une exploitation parallèle. De plus, les nouveaux modèles de puce Virtex II Pro possèdent des serdes (serializer/deserializer) beaucoup plus rapides avec lesquels il serait possible d'implémenter un mécanisme de communication plus léger en terme de cellules logiques et plus rapide.

CHAPITRE 3

SYSTÈMES D'EXPLOITATION

Des plus puissants supercalculateurs aux plus simples systèmes embarqués, l'évolution des systèmes informatiques tend à aller vers une complexité toujours croissante de la composante logicielle. Dans la course au développement de nouveaux produits, l'utilisation de plateformes existantes permet de réduire les temps liés au développement.

Le système d'exploitation est le pont qui relie la couche matérielle à la couche logicielle. Cette couche permet de présenter une vue abstraite du matériel aux applications. Le choix d'un système d'exploitation est donc critique dans la conception d'un système. Il doit y avoir un équilibre entre les ressources consommées par le système d'exploitation et les services qu'il offre.

Dans le cadre d'une grappe de calcul de haute performance, ce qui est habituellement recherché est explicité dans son nom : la performance. Il faut donc choisir judicieusement le système d'exploitation afin de pouvoir obtenir le maximum de performance du matériel tout en sacrifiant le minimum de ressources.

Gentoo Linux est parfaitement indiqué pour ce genre de tâche car cette distribution allie sans compromis performance et flexibilité. En effet, puisque cette distribution s'installe généralement à partir des sources des paquetages, il est possible d'optimiser leur génération afin de mieux correspondre à l'architecture matérielle pour laquelle ils se destinent.

Pour ceci, Gentoo Linux utilise des scripts détaillant les étapes pour la configuration

et l'installation des logiciels qui composent le système. Ces scripts, ou ebuilds, sont contenus dans un arbre de développement qui est mis à jour plusieurs fois par jour.

Adelie Linux maintient plusieurs arbres de développement parallèles pour Gentoo Linux. On peut donc retrouver dans ceux-ci différents paquetages en relation avec les grappes de calcul, les outils de développement parallèles et tout autre paquetage utile au calcul de haute performance. La figure 3.1 présente l'arbre de développement principal de Gentoo Linux ainsi que les branches offertes par Adelie Linux.

Gentoo Linux possède également un mécanisme d'initialisation hautement flexible qui permet l'intégration de grappes de calcul sans disques. Des extensions à ce mécanisme permettent de supporter un grand nombre de configurations concurrentes.

Bien que hautement configurable, Gentoo Linux s'adresse encore à un marché dit de PC et non de systèmes embarqués. Pour l'intégration de ceux-ci, l'outil *PTXdist* permet de générer des environnements GNU/Linux minimaux. En ajoutant les fichiers de configurations et scripts nécessaires pour avoir un système de base, on obtient une micro-distribution spécialement optimisée pour les systèmes limités en ressources.

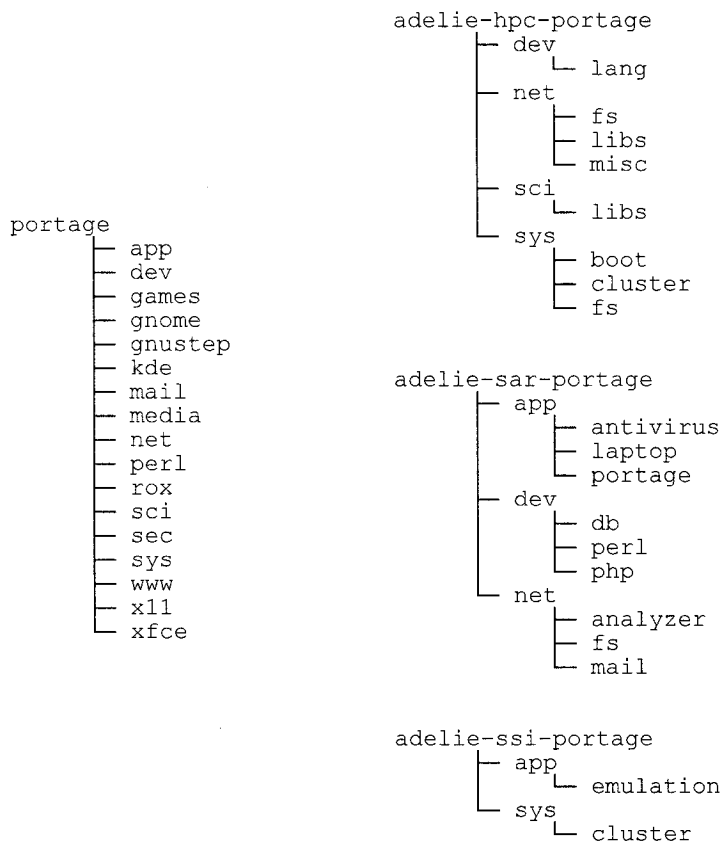

Figure 3.1 Détails des arbres de développement d'Adelie Linux.

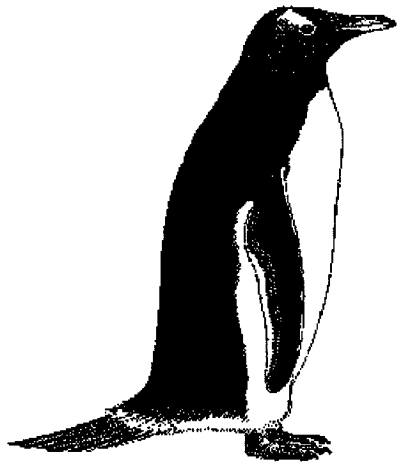

Figure 3.2 Manchot papou *(Pygoscelis papua, Gentoo Penguin).*

3.1 Gentoo Linux

Le manchot papou (Pygoscelis papua, Gentoo Penguin) est reconnu pour sa grande timidité parmi toutes les espèces de manchots. Qu'à cela ne tienne, il est aussi l'oiseau nageur le plus rapide, toute catégorie confondue. C'est pour sa vitesse que Daniel Robbins a choisi en 2002 le manchot papou pour représenter sa nouvelle distribution GNU/Linux.

Gentoo Linux se veut différent des autres distributions en venant rompre avec la tradition. Là où la plupart des distributions se distinguent plus par leurs paquetages et leurs outils de configuration, Gentoo Linux tente d'améliorer la structure fondamentale. Trois caractéristiques principales séparent donc Gentoo Linux des autres distributions.

Gentoo Linux s'installe à partir d'un environnement Linux et ce manuellement. Exit les interfaces graphiques bien intégrées. Une invite de commande en mode texte est tout ce qui est requis pour installer Gentoo Linux. Il est donc possible de l'installer à partir d'une autre distribution GNU/Linux, et ce sans nuire au fonc-

tionnement de celle-ci. Ou encore, il est possible d'utiliser un cdrom comprenant un système Gentoo Linux fonctionnel pour l'installer.

Gentoo Linux n'utilise pas de paquetages pré-compilés. Un mécanisme permettant la compilation des paquetages à partir des sources est utilisé. Des scripts automatisant le téléchargement, la configuration, la compilation et l'installation des paquetages sont utilisés. Ces scripts sont maintenus dans ce que Gentoo Linux appelle le Portage Tree, qui contient l'ensemble de ces scripts. Le Portage Tree est mis à jour à continuellement, il n'y a donc pas de version de Gentoo Linux.

Gentoo Linux rompt avec la tradition et n'utilise pas le processus d'initialisation System V. Un mécanisme plus flexible permet de gérer dynamiquement les dépendances des scripts d'initialisation. De plus, Gentoo Linux n'est pas limité aux huit niveaux d'initialisation classiques. Ces niveaux sont en nombre illimités sous Gentoo et peuvent prendre des noms complexes.

Ces différentes composantes font de Gentoo Linux une distribution hautement configurable et particulièrement optimisée.

3.1.1 Portage Tree

Le Portage Tree de Gentoo Linux est composé d'un ensemble de scripts. Ces scripts, ou ebuilds, permettent de télécharger, configurer, compiler et installer des paquetages à partir des sources. Écrits en langage de scriptage Bash, ces scripts permettent d'automatiser toutes ces étapes de manière transparente à l'utilisateur.

Les ebuilds sont conservés de manière hiérarchique dans ce que l'on appelle le Portage Tree. Celui-ci peut être synchronisé à l'aide d'utilitaires simples à un des nombreux miroirs disponibles. Ces miroirs sont mis à jour à toutes les quatre

heures. Il est donc possible de maintenir un système à jour sans jamais avoir à réinstaller complêtement le système d'exploitation.

Le concept derrière le Portage Tree n'origine pas de Gentoo Linux. Il s'agit en fait d'un mécanisme que l'on retrouve sur diverses distributions BSD. Sous OpenBSD, celui-ci porte le nom de Ports Collection; sous NetBSD on le retrouve sous pkgsrc Package Collection. Tous descendent en fait des Ports Collection de FreeBSD.

L'utilisation de ce mécanisme comporte de nombreux avantages par rapport aux distributions classiques. En effet, celles-ci offrent habituellement une seule version d'un paquetage particulier. Que l'on se trouve sur un simple Pentium ou encore sur un puissant Xeon, tous ces paquetages auront étés compilés pour le plus petit dénominateur commun. Donc, on ne retrouve pas d'optimisation spécifique à l'architecture du processeur.

Sous Gentoo Linux, il est possible de spécifier les options d'optimisation à utiliser lors de la compilation des paquetages. Ainsi, il est possible d'optimiser spécifiquement pour l'architecture sur laquelle s'exécuteront ses applications.

De plus, les paquetages binaires doivent satisfaire la plus grande majorité des configurations possibles. Donc impossible d'avoir une version texte uniquement de Xemacs, le support X Window y est compilé car il faut pouvoir satisfaire à tous. Le Portage Tree, puisqu'il permet de compiler à partir des sources et ce localement, permet donc de configurer tous les paramètres optionnels des différents paquetages.

À l'aide d'une liste de mots clés nommés Use Flags, il est donc possible de spécifier les composantes optionnelles à inclure dans chaque paquetage. Ces Use Flags peuvent être définis globalement ou encore localement, pour chaque paquetage. Cela permet donc d'avoir seulement les options qui sont d'intérêt pour le système local.

Il est ainsi possible d'avoir des applications plus petites car ne comprenant pas tout le code optionnel inutile, et plus performantes puisqu'optimisées spécifiquement pour la plateforme courante.

Il y a bien sûr un désavantage à tout ceci. Il est plus long d'installer un paquetage sous Gentoo Linux que sous une autre distribution. On peut donc sauver quelques heures à l'installation en choisissant une autre distribution. Mais les pertes de performances associées à l'utilisation de paquetages binaires pré-compilés peuvent se solder en des pertes de temps beaucoup plus importantes.

L'outil principal pour gérer ce mécanisme se nomme *emerge* sous Gentoo Linux. Celui-ci peut être utilisé pour synchroniser la mise à jour du Portage Tree. Il est aussi utilisé pour installer de nouveaux paquetages. Dans ce cas, il gère automatiquement la résolution des dépendances et ce dynamiquement. C'est-à-dire que si un paquetage est installé en utilisant certains Use Flags, ceux-ci peuvent affecter les dépendances de ce paquetage.

Emerge est également utilisé pour mettre à jour les paquetages déjà installés sur le système. Il est donc possible de mettre à jour tous les paquetages qui sont installés, pour lequels une version plus récente est disponible, suite à une mise à jour du Portage Tree. Ou encore, si les Use Flags globaux ont été changés, tous les paquetages affectés par ce Use Flags peuvent être recompilés.

Il est donc fort simple avec *emerge* de mettre à jour un système entier, de façon dynamique, et ce tous les jours. Il suffit donc de synchroniser le Portage Tree, ainsi que tous les autres arbres de développement additionnels se trouvant sur le système, et ensuite de lancer l'installation des nouveaux paquetages. Ces deux étapes fort simples suffisent à maintenir à jour un système et peuvent être facilement automatisées en une tâche journalière.

Il est de plus possible de restreindre le choix des applications jugées critiques. Si par exemple, une version particulière d'un logiciel est nécessaire, il est possible de spécifier ce paquetage ainsi que cette version dans un fichier de configuration afin de masquer toute autre version. Ainsi, si l'on automatise le processus de synchronisation et de mise à jour, on ne risque pas de se retrouver avec des problèmes de compatibilité.

Un autre aspect intéressant du Portage Tree est l'utilisation d'un environnement protégé pour la compilation des paquetages. En effet, un carré de sable est mis en place lors de l'installation de paquetage qui empêche celui-ci de modifier le système à sa guise. Les applications sont donc installées dans un environnement spécial avant d'être intégrées au système.

De cette manière, la liste de tous les fichiers créés lors de l'installation d'un paquetage est conservée par le système. Une désinstallation ultérieure de ce dernier ne laisse pratiquement pas de trace. Comme il arrive pratiquement toujours qu'un administrateur doit installer un logiciel pour lequel aucun paquetage n'est disponible, l'utilisation de l'environnement protégé devient particulièrement intéressant.

Un administrateur n'a donc qu'à rédiger un ebuild pour ce logiciel, dans une section privée du Portage Tree. Ensuite, il lui sera possible d'installer mais surtout de désinstaller à son gré ce logiciel, ce qui peut s'avérer très pratique lorsqu'une nouvelle version du logiciel est disponible.

Ces différents aspects du Portage Tree, le mécanisme d'installation des paquetages sous Gentoo Linux, font de cette distribution une des plus appréciées des administrateurs concernés par la performance de leurs systèmes. De plus, celui-ci permet de maintenir à jour un système de façon journalière et ce presqu'automatiquement. Comme la rédaction d'ebuilds s'avère un exercice simple, il permet l'ajout facile et

sécuritaire de logiciels qui n'y sont pas présents.

3.1.2 Gentoo Init

Sous un système UNIX, GNU/Linux ou BSD, le processus d'initialisation est le mécanisme qui prend en charge le démarrage des différents services. Initié immédiatement après l'exécution du noyau, ce processus gère le démarrage et, lors de la fermeture du système, l'arrêt des différents logiciels qui doivent s'exécuter en arrière plan. Un mécanisme permet de gérer automatiquement les dépendances qui peuvent exister entre certains services.

La plupart des systèmes GNU/Linux utilisent encore un processus d'initialisation basé sur celui d'UNIX System V, développé en 1983 par AT&T. Ce mécansime fort simple était parfaitement adapté aux systèmes de l'époque, où les ressources étaient limitées et par conséquent le nombre et la complexité des services particulièrement restreints.

Gentoo Linux est une des rares distributions GNU/Linux à ne pas utiliser un processus d'initialisation de type System V. En effet, celui-ci a été remplacé par un mécanisme plus flexible mais qui en revanche demande un peu plus de traitement de la part du système.

À chaque service est associé un script d'initialisation. Les scripts d'initialisation sont tous conservés dans le répertoire `/etc/init.d` sous Gentoo Linux ou sous `/etc/rc.d/init.d` pour la plupart des autres distributions. Ces scripts possèdent une interface commune, c'est-à-dire qu'ils acceptent des arguments standard et retournent des codes d'erreurs définis. Les arguments mutuellement exclusifs possibles sont :

- `start` : Démarre le service.
- `stop` : Arrête le service.
- `status` : Affiche l'état (démarré/arrêté) du service.
- `restart` : Si le service est démarré, arrête puis redémarre ce service, tout en respectant les dépendances.
- `reload` : Force une relecture des fichiers de configurations sans interrompre le service.

Ces arguments ne doivent pas tous être nécessairement implémentés. Par exemple, de nombreux services ne supportent pas la relecture des fichiers de configuration. Traditionnellement, une relecture des fichiers de configuration est déclenchée par la réception d'un signal SIGHUP. Il est possible que certains services ne supportent pas ce mécanisme ou tout simplement qu'ils ne possèdent pas de fichiers de configuration.

Les codes de retour sont eux aussi définis. Habituellement, le code de retour du script devrait être celui du dernier appel échoué ou de la dernière commande dans le cas d'un succès. Il est du ressort du script de vérifier ces codes afin de retourner un code d'erreur approprié, dans le cas d'un script comprenant plusieurs commandes. Voici les deux cas de codes de retour possibles:

- 0 : Succès. Dans le cas d'un démarrage, le service est considéré comme démarré, dans le cas d'un arrêt, le service est considéré comme arrêté.
- >0 : Échec. Le service n'a pas pu être démarré ou arrêté correctemnt. Dans le cas d'un démarrage, le service est toujours considéré comme arrêté, dans le cas d'un arrêt, le service est toujours considéré comme démarré. Les services qui dépendent de ce dernier ne pourront être démarrés ou arrêtés correctement.

Les processus d'initialisation de type System V comprennent typiquement huit niveaux d'initialisation bien que onze niveaux soient possibles. Les noms de ces niveaux sont fixes, soit 0 à 6 et S. Les états du système auxquels correspondent ces niveaux peuvent varier d'une distribution à une autre. Les niveaux appelés 0, 1, 6 et S sont pratiquement standard sur tous les systèmes. Voici les huit niveaux communs et leur emploi sous SuSE Linux et RedHat Linux :

- 0 : Arrêt.
- 1 : Mode utilisateur unique.
- 2 : Mode multi-utilisateurs avec interface texte sans réseau (SuSE). Non utilisé (RedHat).
- 3 : Mode multi-utilisateurs avec interface texte et réseau (SuSE et RedHat).
- 4 : Non utilisé (SuSE et RedHat).
- 5 : Mode multi-utilisateurs avec interface graphique et réseau (SuSE et RedHat).
- 6 : Redémarrage.
- S : Mode utilisateur unique.

Sous Gentoo Linux, ces niveaux ne sont pas limités en nombre ni en noms. Un paramètre du noyau, `softlevel=<runlevel>`, permet de spécifier lesquel est employé lors de l'initialisation. Cela permet donc sur une même image de système d'offrir une infinité de configurations différentes. Par défaut, on retrouve sur un système Gentoo quelques niveaux communs, bien qu'il soit possible d'en redéfinir d'autres :

- `boot` : Initialisation commune.
- `default` : Mode multi-utilisateurs avec réseau.
- `nonetwork` : Mode multi-utilisateurs sans réseau.
- `single` : Mode utilisateur unique.

La résolution des dépendances au sein d'un même niveau diffère grandement entre le processus d'initialisation de Gentoo et celui du System V.

Dans le cas d'un processus de type System V, les dépendances sont spécifiées dans l'entête du script d'initialisation. Ce mécanisme n'est pas universel au sein de toutes les distributions GNU/Linux bien que le projet LSB (Linux Standard Base) tente de standardiser le tout. Le mécanisme proposé par le LSB définit les dépendances suivantes :

- `Required-Start` : Liste des services qui doivent être actifs afin de démarrer celui-ci.
- `Required-Stop` : Liste des services qui doivent être actifs afin d'arrêter celui-ci.
- `Should-Start` : Liste des services qui, s'ils sont présents, doivent être actifs afin de démarrer celui-ci.
- `Should-Stop` : Liste des services qui, s'ils sont présents, doivent être actifs afin d'arrêter celui-ci.

En général, un service requis au démarrage est requis à l'arrêt, les services sont donc arrêtés dans l'ordre inverse où ils ont été démarrés. Pour cette raison, Gentoo

Linux n'offre pas de dépendances explicitement différentes au démarrage et à l'arrêt. De plus, celles-ci ne sont pas des commentaires statiques mais bien des appels de fonctions. En plus des dépendances de type prérequis, Gentoo introduit les dépendances de types postrequis. Voici les dépendances possibles :

- `need` : Liste des services qui doivent être actifs afin de démarrer celui-ci.
- `use` : Liste des services qui, s'ils sont présents, doivent être actifs afin de démarrer celui-ci.
- `before` : Liste des services, s'ils sont présents, avant lesquels celui-ci doit être démarré.
- `after` : Liste des services, s'ils sont présents, après lesquels celui-ci doit être démarré.

À chaque niveau d'initialisation correspond un répertoire particulier. Pour insérer un service dans un niveau particulier, un lien symbolique est créé dans le répertoire correspondant qui pointe vers le script associé au service.

Sous un processus de type System V, les dépendances sont résolues statiquement à l'ajout ou au retrait d'un service. Le nom que prendra le lien symbolique indiquera son ordre d'exécution. Typiquement, ils se présentent sous la forme `S<priority><service>` pour le démarrage ou `K<priority><service>` pour l'arrêt.

Sous Gentoo Linux, les dépendances seront calculées au démarrage à partir des services présents dans le niveau d'initialisation. Il est donc impossible d'introduire des erreurs dans l'ordre d'initialisation.

Le processus d'initialisation de Gentoo Linux est donc plus flexible et plus robuste que celui proposé en 1983 par UNIX System V. La possibilité d'avoir un nombre ilimité de niveaux d'initialisation permet de supporter une multitude de configurations différentes à partir d'une même image de système. De plus, îl s'avère beaucoup moins cryptique de parler du niveau `nonetwork` que du niveau `rc2`.

3.2 Adelie/PTX

Les distributions GNU/Linux classiques ne sont pas destinées à des environnements où les ressources sont limitées. C'est également le cas de la distribution Gentoo Linux. Des projets sont en cours permettant de l'adapter aux systèmes embarqués ainsi qu'aux PDA.

La solution développée permet d'intégrer des mécanismes de configuration similaires à ceux de Gentoo Linux tout en préservant une taille minimale. Basé sur l'outil *PTXdist*, l'environnement Adelie/PTX permet ainsi d'avoir une configuration qui est compatible avec les mécanismes d'Adelie/SSI.

Gentoo Linux permet grâce au Portage Tree de spécifier quelles composantes optionnelles seront configurées et intégrées dans chaque paquetage installé. Cela permet d'obtenir des gains en terme d'espace disque utilisé mais aussi au point de vue de la mémoire vive nécessaire pour les exécuter.

Il est donc possible d'avoir un système Gentoo Linux optimisé pour la majeure partie du système. Reste tout de même que le système de base de Gentoo Linux, sans ses paquetages optionnels, demeure d'une taille imposante. Faire fonctionner un système avec moins de 32 Mo de mémoire vive reste un défi difficile à relever.

3.2.1 Embedded Gentoo, TinyGentoo et Adelie/PTX

Il existe des projets qui tentent d'adaper Gentoo Linux à des systèmes plus compacts. Embedded Gentoo est un de ceux-ci. Grâce à un support pour la compilation inter-plateforme, un système Gentoo Linux tiers peut gérer la compilation et l'installation de paquetages pour un système embarqué. Une version allégée du système de base, `baselayout-lite`, est aussi fournie.

Malheureusement, cette version n'est pas compatible avec les scripts d'initialisation traditionnels de Gentoo Linux. Il n'est donc pas possible d'intégrer une distribution de type Embedded Gentoo dans un système Gentoo Linux.

Un autre projet, TinyGentoo, vise plutôt à adapter Gentoo Linux à des systèmes dont la taille des disques est réduite. Il permet ainsi d'obtenir un système fonctionnel occupant moins de 64 Mo d'espace disque. Malheureusement, ce projet tente plutôt de réduire la taille du système que la complexité de celui-ci.

Une solution combinant certains aspect des deux projets a été développée. Il s'agit d'une distribution qui tente de réduire la taille requise du système, tant sur disque qu'en mémoire. De plus, cette distribution utilise des scripts d'initialisation dont le fonctionnement est compatible avec ceux que l'on retrouve dans une distribution Linux standard.

Cette distribution, nommée Adelie/PTX, utilise l'outil de configuration *PTXdist*. Cet outil permet de générer des systèmes GNU/Linux basés sur *BusyBox*. De plus, il intègre l'outil *Crosstool* qui permet la compilation des outils de développement GNU classiques inter-plateforme.

3.2.2 PTXdist

Développé par la firme allemande Pengutronix, l'outil *PTXdist* permet de configurer et de générer un ensemble d'applications afin d'assembler un système GNU/Linux de base. Il est principalement destiné aux systèmes embarqués où tant la mémoire que l'espace disque sont des ressources limitées.

Grâce à *PTXdist*, il est possible de créer des outils de compilation inter-plateforme et d'utiliser ceux-ci pour générer des exécutables formant la base d'un système

minimal. Il est donc possible de compiler un système de base comprenant un noyau Linux, des librairies de base ainsi que les utilitaires minimaux.

De plus, il est possible d'inclure les fichiers de configurations additionnels pour former un système fonctionnel. *PTXdist* offre de telles arborescences pour différents types de plateformes existantes. Il est possible d'inclure ses propres fichiers si ceux-ci ne correspondent pas aux besoins.

Afin de créer les outils de compilation, *PTXdist* utilise les excellents scripts de génération d'environnements de développement inter-plateforme *Crosstool*. Développé par Dan Kegel, *Crosstool* est un ensemble de scripts permettant de compiler une chaîne entière d'outils de développement pour une platefrome matérielle tierce.

Il est donc possible de compiler les outils GNU binutils, glibc et gcc pour une plateforme précise. Des scripts sont donnés pour plusieurs processeurs, ainsi que pour différentes versions de glibc et de gcc. Des rustines sont incluses lorsque certaines configurations posent problème, ce qui est très fréquent en dehors de l'architecture x86.

Une fois les outils de développement inter-plateforme créés à l'aide de *Crosstool*, *PTXdist* offre la possibilité de configurer et de compiler des librairies du système pour la plateforme de destination. Il sera ensuite possible de compiler *BusyBox*, une véritable boîte à outils extrêmement compacte.

BusyBox est un exécutable compact qui peut émuler le fonctionnement de bon nombre d'applications communes aux systèmes GNU/Linux. En fait, il arrive à lui seul à remplacer les coreutils de GNU, ces utilitaires de base, comprenant les utilitaires de fichiers, de textes et de shell. Il est donc théoriquement possible d'avoir un système fonctionnel complet uniquement avec *BusyBox*.

L'ensemble de ces outils forment la base d'un système GNU/Linux. Pour avoir un système complet, il faut ajouter le noyau Linux approprié ainsi que les fichiers de configuration et les scripts nécessaires à l'initialisation et au fonctionnement du système.

3.2.3 TimeSys Linux

Le noyau Linux est probablement le logiciel qui a été porté au plus grand nombre de plateformes. On ne compte pas moins de 24 architectures, sans compter les sous-architectures regroupées sous les bannières de celles-ci. Il reste néanmoins bon nombre de plateformes, particulièrement dans le domaine des systèmes embarqués, qui nécessitent un travail d'adaption pour y faire rouler Linux.

La firme TimeSys offre un vaste éventail de noyaux Linux spécifiquement adaptés pour divers systèmes embarqués. Ces noyaux présentent en plus dans certains cas des améliorations pour permettre une utilisation de ces noyaux dans un contexte temps-réel.

Le noyau TimeSys Linux a été porté pour la plateforme progicielle se retrouvant sur les cartes de la famille AP100 d'Amirix. Celle-ci intègre des pilotes pour tout le matériel se retrouvant sur cette plateforme, y compris les adaptateurs réseaux. Il est donc possible d'utiliser ce noyau sans besoin de le modifier davantage.

Le noyau utilisé sur les noeuds embarqués est un noyau de la lignée 2.4. Il s'agit spécifiquement du noyau 2.4.18-timesys-4.0.354. D'autres versions existent mais ne supportent pas la plateforme gigabit d'Amirix.

Ce noyau supporte l'utilisation d'une image racine via NFS. C'est dans ce contexte que le noyau est utilisé. L'image racine réside sur le serveur et est partagée par

tous les noeuds embarqués. Malheureusement, les interfaces réseaux présentes sur ces noeuds ne supportent pas la norme PXE. Le noyau doit donc résider dans la mémoire flash des plateformes embarquées.

Pour charger le noyau, l'application résidente *ppcboot* est employée. Il s'agit d'un outil qui permet de gérer les différentes mémoires disponibles sur la plateforme. C'est donc cet outil qui se charge de copier en mémoire vive le noyau résident en mémoire flash puis de l'éxécuter.

3.2.4 Adelie/PTX

Afin d'avoir un système GNU/Linux fonctionnel, en plus du noyau, des librairies et des utilitaires de base, il faut un certain nombre de fichiers supplémentaires. Ces fichiers se divisent en deux catégories, soit les fichiers de configuration et les scripts d'initialisation.

Au nombre des fichiers de configuration on retrouve la liste des utilisateurs, la liste des groupes, la liste des partitions, la liste des noms d'hôtes et plusieurs autres.

Les scripts quant à eux s'articulent autour du script principal, `/sbin/rc`. C'est celui-ci qui est invoqué lors de l'initialisation. Dans un système minimal, ce script pourrait lancer lui-même tous les services nécessaires. Afin d'avoir plus de flexibilité, ces services sont habituellement démarrés dans des scripts qui leurs sont propres. Ainsi, il est plus facile d'ajouter ou d'enlever un service particulier.

L'ensemble de cette configuration, intitulé Adelie/PTX, est conçu afin de s'intégrer dans un système à image hybride basé sur Adelie/SSI. Ainsi, il utilise un format de script d'initialisation et de configuration compatible avec celui de Gentoo Linux. Toutes les fonctionnalités n'y sont pas présentes.

Un ensemble de services est inclu afin d'assurer le bon fonctionnement de la plateforme embarquée.

3.2.5 adelie-ssi

Le fonctionnement du script de mise en place de l'environnement des noeuds embarqués est similaire à celui des noeuds hôtes. Puisque la mémoire vive est limitée sur les noeuds embarqués, il n'est pas possible d'utiliser le mécanisme expérimental par superposition.

Le script d'initialisation `adelie-ssi` est donc responsable de créer les répertoires locaux en mémoire vive, d'y copier les fichiers des répertoires correspondant de l'image racine du système de fichier et d'y créer les liens symboliques permettant d'avoir une configuration unique sur chaque noeud.

3.2.6 bootmisc

Le service bootmisc est utilisé uniquement pour monter le système de fichier proc.

3.2.7 clock

La plateforme embarquée ne possède pas d'horloge temps réel propre. Donc, l'heure ainsi que la date courante sont perdues lorsque les noeuds sont éteints ou lors d'un redémarrage. Lors de l'initialisation du noyau, l'horloge logicielle de Linux est initialisée à l'époque, qui pour Linux est minuit le 1er janvier 1970.

Afin de synchroniser l'horloge, différents mécanismes sont possibles. Le protocole NTP permet une telle synchronisation. Malheureusement, le démon *ntpd* est rela-

tivement lourd pour les ressources limitées de la plateforme embarquée.

Un mécansime simple est alors utilisé. L'heure d'une machine tierce est obtenue par le biais de SSH et de la commande *date*. Cette heure, ordonnée correctement, est ensuite utilisée comme argument à la commande date sur la plateforme embarquée. De cette manière, l'horloge locale se trouve synchronisée à l'horloge d'une machine tierce.

L'adresse de la machine avec laquelle doit se synchroniser le noeud embarqué peut être spécifiée dans le fichier de configuration du script d'initialisation. Dans ce cas-ci, l'horloge est synchronisée avec le serveur.

De plus, le noyau Linux maintient une horloge en temps UTC. Pour utiliser un fuseau horaire local, un fichier de définition de zone doit être utilisé. Dans ce cas, il s'agit du fuseau horaire Canada Eastern.

3.2.8 locamount

Bien que l'image racine soit fournie par le serveur, il est possible de monter des systèmes de fichiers locaux ou encore réseaux. Le service localmount a pour but de charger tous les systèmes de fichiers présents dans le fichier de configuration `/etc/fstab` qui ne sont pas présentement montés.

3.2.9 net

La configuration des interfaces réseaux est propre à chaque noeud. Le serveur génère celle-ci à partir de la configuration centrale d'Adelie/SSI. Pour pouvoir les utiliser, il faut que le script d'intialisation accepte les fichiers de configuration du format de Gentoo Linux.

L'utilitaire `runscript`, qui invoque ensuite le script `runscript.sh`, détecte les scripts d'initialisation de type `net`. L'extension du nom du script est utilisé pour déterminer le nom de l'interface réseau à configurer. La configuration associée à cette interface est ensuite passée en argument à l'application `ifconfig`.

3.2.10 ypcat

Afin de partager la liste des utilisateurs et des groupes avec le serveur, différents protocoles peuvent être utilisés. NIS est celui employé sur les noeuds hôtes de la grappe. Il est également possible d'utiliser LDAP dans une de ses diverses implémentations.

Les démons et utilitaires sont eux aussi de taille importante. Afin de minimiser l'espace nécessaire, un mécanisme similaire à la synchronisation de l'horloge est employé.

Lors du démarrage des noeuds embarqués, les listes des utilisateurs et des groupes sont synchronisées avec celles du serveur NIS. Une requête est donc effectuée à l'aide de l'utilitaire *ypcat* accédé via ssh sur un noeud hôte.

Les listes obtenues ne contiennent pas l'ensemble des utilisateurs du système. En effet, les utilisateurs tel que root ne sont habituellement pas exportées par NIS. Une liste locale minimale existe donc sur chaque noeud embarqué. Cette liste est filtrée pour éliminer tous les utilisateurs et groupes susceptibles de se retrouver dans les listes de NIS.

La liste d'utilisateurs obtenue par NIS ainsi que la liste locale sont filtrées pour s'assurer que le shell de l'utilisateur est bien *ash* et non *bash* par exemple. En effet, seul ash est présent sur les noeuds embarqués.

Les listes sont ensuite fusionnées sur les noeuds embarqués. Ainsi les listes des utilisateurs et des groupes sont resynchronisées avec le serveur à chaque invocation du script.

3.2.11 ssh

Il existe différents mécanismes permettant aux utilisateurs de se brancher sur une machine distante. La plupart des protocoles non-encryptés tel que Telnet ou RSH sont amenés à disparaitre progressivement. En effet, la transmision des mots de passe en texte clair les rend particulièrement vulnérables aux intercepteurs de paquets. Le protocole SSH offre une sécurité accrue. Un système de clés est utilisé pour encrypter les communications.

La couche hôte utilise OpenSSH pour permettre les branchements entre les noeuds. Pour faciliter l'utilisation d'applications parallèles, un mécanisme de validation par clés d'hôtes est utilisé au lieu de mot de passe. L'authentification se fait donc à partir de la clé globale de la machine de source.

Afin d'intégrer les noeuds embarqués au sein de la grappe de calcul, OpenSSH y est installé. La configuration y est exactement la même que sur la couche hôte.

CHAPITRE 4

ADELIE LINUX

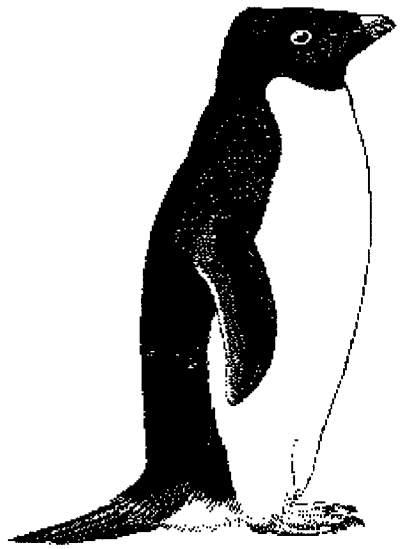

Figure 4.1 Manchot Adélie *(Pygoscelis adeliae, Adelie Penguin).*

Le manchot Adélie (Pygoscelis adeliae, Adelie Penguin), nommé ainsi d'après la femme de l'explorateur français Dumont d'Urville, est une des trois espèces de manchots du genre Pygoscelise. Les deux autres membres de ce genre sont le manchot à jugulaire (Pygoscelis antartica, Chinstrap Penguin) et le manchot papou (Pygoscelis papua, Gentoo Penguin). Cet oiseau nageur de l'Antarctique est une des espèces les plus nombreuses de manchots.

En 2002, au sein du groupe de CHP (calcul de haute performance) du CERCA (centre de recherche en calcul appliqué), ce nom apparut pour la première fois en association avec les grappes de calculs. Ce nom fut donné à la première tentative réussie de grappe de calculs sans disque basé sur Gentoo Linux. Le mécanisme employé sera par la suite renommé Adelie/SSI (Single Image system).

4.1 Adelie/SAR, Adelie/HPC et Adelie/SSI

Adelie Linux est un étendard pour une suite d'outils logiciels développés pour faciliter le déploiement et la gestion de systèmes basés sur Gentoo Linux. Actuellement, il existe trois branches principales à ce projet :

- Adelie/SAR : Ressources pour la gestion de systèmes (System Administration Resources)
- Adelie/HPC : Calcul de haute performance (High Performance Computation)
- Adelie/SSI : Image unique de système de fichiers (Single System Image)

Chaque branche possède son propre arbre de portage dans lequel sont maintenus à jour les scripts de déploiement pour des applications tierces ne faisant pas partie ou différent de l'arbre de portage officiel de Gentoo Linux. De plus, une multitude de sous-projets locaux sont contenus sous ces bannières.

Sous la branche Adelie/SAR sont regroupés les projets et les scripts ayant rapport avec l'administration et la gestion de systèmes basés sur Gentoo Linux. Ces outils peuvent habituellement être utilisés dans le contexte de stations de travail, de serveurs ou même de grappes de calcul.

La branche Adelie/HPC quant à elle regroupe les projets et les scripts ayant rapport avec le calcul de haute performance. On y retrouve donc les scripts de déploiement pour un grand nombre d'applications et de librairies utilisées sur les systèmes parallèles. On y retrouve également tous les projets de développement d'applications parallèles ou en venant en aide à celles-ci.

La dernière branche contient quant à elle le projet initialement nommé Adelie Linux mais ramené aujourd'hui à une branche de ce dernier, soit Adelie/SSI. Adelie/SSI

regroupe ainsi tous les scripts, fichiers de configurations, applications et autres permettant le déploiement rapide et la gestion centralisée de grappes de calcul sans disque *(diskless)* partageant une seule et même image du système de fichiers.

Adelie Linux est donc un projet parapluie regroupant trois catégories distinctes de scripts de déploiement et d'applications. Ces trois catégories sont Adelie/SAR pour l'administration et la gestion de systèmes, Adelie/HPC pour l'exploitation des systèmes de calculs de haute performance et finalement Adelie/SSI pour le déploiement et la gestion de grappes de calcul partageant une image unique de système de fichiers.

4.2 Adelie/SSI ou image de système de fichiers unique

Adelie/SSI est un ensemble de logiciels *(toolkit)* permettant le déploiement rapide et la gestion centralisée de grappes de calcul et de réseaux de postes de travail ou Now (Network of Workstations) sans disque partageant une seule et même image du système de fichiers. Cet ensemble de logiciels s'intègre dans un système basé sur Gentoo Linux ou comportant un mécanisme de configuration compatible.

Dans ce contexte, une grappe de calcul est un ensemble de systèmes, comprenant un serveur ainsi que un ou plusieurs noeuds, reliés entre eux par un ou plusieurs mécanismes de communications à l'usage exclusif de l'ensemble de ces systèmes. Ce matériel peut être du matériel informatique générique ou encore du matériel de haute performance spécialisé.

Il n'y a typiquement pas d'accès interactif aux noeuds, si ce n'est qu'à des fins de maintenance. L'utilisation de la grappe de calcul s'effectue via le serveur qui fait office de système frontal. Sur ce frontal sont lancées les applications qui s'exécuteront sur un ou plusieurs noeuds. Le serveur peut également agir comme passerelle pour relier les noeuds et le monde extérieur.

Un réseau de postes de travail est une version moins stricte d'une grappe de calcul. Il s'agit là aussi d'un ensemble de systèmes, comprenant ou non un serveur dédié, reliés entre eux par un mécanisme de communication qui n'est pas nécessairement exclusif à l'ensemble des postes de travail.

Le mode d'utilisation principal de ces postes est interractif, à l'exception peut-être du serveur dédié s'il y a. Les utilisateurs n'exploitent en général que leur propre poste de travail. Les applications y sont donc exécutées localement. Les postes peuvent se retrouver derrière un pare-feu mais ont habituellement accès au monde

extérieur.

Adelie/SSI permet donc le déploiement rapide de grappes de calcul et de réseaux de postes de travail. Pour permettre ce déploiement, c'est-à-dire l'ajout d'un nouveau noeud ou poste de travail ou encore l'installation ou la mise à jour de logiciels sur ce dernier, une architecture sans disque local est utilisée.

Pour permettre à un noeud ou à un poste de travail de fonctionner sans disque, deux choses sont importantes. Le système en question doit pouvoir obtenir son noyau par réseau. Il doit ensuite pouvoir accéder à un système racine de fichiers, partagé ou non, également par réseau. Ainsi aucune image locale n'est requise, l'image réside plutôt sur le serveur ou le poste de travail maître.

Le système se partage donc une seule et même image de système unique de fichiers. Il s'agit en fait de l'image racine complète du serveur ou du poste de travail maître qui est partagée avec les noeuds ou les postes de travail. Il n'y a donc qu'une seule image de système à maintenir, celle se trouvant sur le serveur ou le poste maître. L'installation d'un logiciel se verra ainsi refléter sur l'ensemble des noeuds ou postes que comprend le système.

Adelie/SSI centralise la gestion de grappes de calcul et de réseaux de postes de travail. En effet, la configuration complète de la grappe de calcul ou du réseau de postes de travail est maintenue dans un fichier central. Ce fichier contient les éléments caractéristiques de la grappe ou du réseau de postes et peuvent être spécifiques à chaque noeud ou poste, à un sous-ensemble de ceux-ci ou encore à leur ensemble.

Un système de configuration modulaire permet ensuite de reconfigurer les services installés à partir des informations contenues dans le fichier central. Le fichier de configuration principal d'Adelie/SSI agit donc comme une banque de données uni-

forme. Les modules de configuration viennent puiser leurs informations dans cette banque de données pour en générer des fichiers de configuration dans le format propre à l'application désirée.

Pour l'instant, Adelie/SSI s'intègre dans un système basé sur Gentoo Linux. Des différences fondamentales entre Gentoo Linux et les autres distributions GNU/Linux limitent son utilisation directe sur celle-ci. Des mécanismes pour adapter Adelie/SSI aux autres distributions ont été proposés.

Adelie/SSI peut par contre gérer la configuration d'une seconde (ou plus) image de système, à condition que les formats des fichiers de configuration soient compatibles avec ceux de Gentoo Linux. C'est le cas de l'image de système des noeuds embarqués sur la grappe de calcul hétérogène Thakur.

Adelie/SSI est particulièrement adapté aux grappes de calcul et aux réseaux de postes de travail sans disque. Cette trousse d'outils permet de déployer rapidement de nouveaux noeuds ou postes de travail. De par son image unique de système de fichiers, l'installation de nouveaux logiciels sur l'ensemble du système est simplifiée. De plus, le fichier de configuration central ainsi que les modules de configuration permettent une gestion simplifiée du système.

4.2.1 Principe de fonctionnement

Adelie/SSI met en place plusieurs mécanismes pour permettre à un ensemble d'ordinateurs de partager une image unique de système de fichiers. Pour charger le système d'exploitation sur les noeuds, un processus de démarrage, via réseau, est employé. Une fois le système d'exploitation chargé et avant le démarrage des différents services, des modifications doivent être apportées à l'environnement logiciel pour permettre au système de fonctionner correctement.

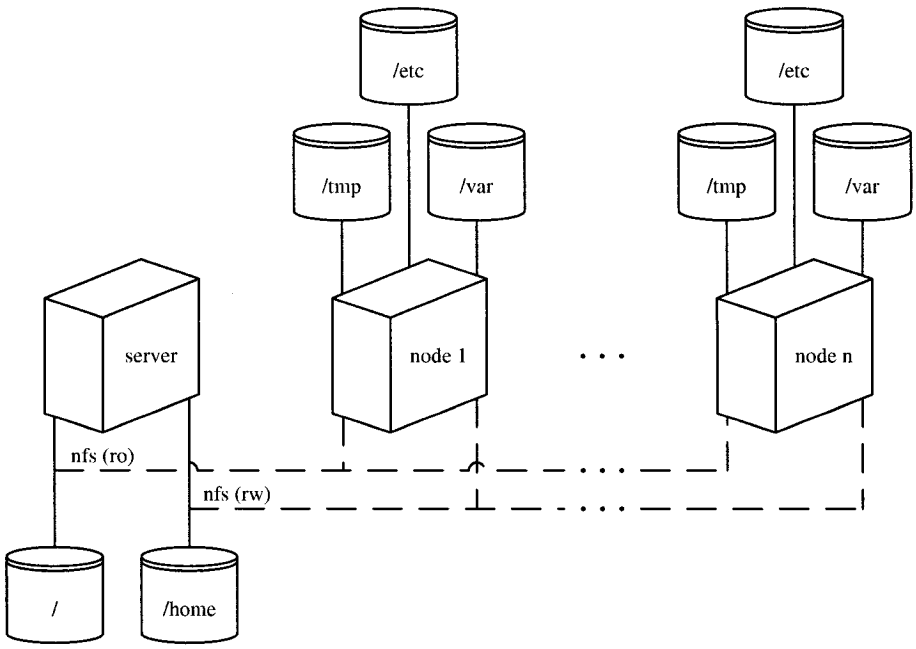

Figure 4.2 Systèmes de fichiers à image unique.

Le nom Single System Image ou système à image unique peut porter à confusion. Il a été adopté au début du projet alors que ce terme était encore mal défini au sein de la communauté. Avec le temps, son sens s'est précisé en divergeant toutefois de sa signification originelle.

On entend traditionnellement par système à image (WALKER, 2003) unique, un système composé de plusieurs noeuds partageant un espace de processus commun, ainsi que des ressources associées, tel que les descripteurs de fichiers. Cet espace de processus commun est ce que l'on nomme image unique de système. Différents exemples de systèmes à image unique, se conformant à un niveau variable à ce paradigme, existent sous Linux: OpenMosix, OpenSSI, Scyld.

Sous Adelie/SSI, ce qui est partagé est plutôt l'image racine du système de fichiers. Cette image, qui est utilisée par l'ensemble du système, réside sur le disque local du serveur. Tant le serveur que les noeuds sont agnostiques quant au choix de

l'implémentation matérielle (IDE, SATA, SCSI, RAID) que logicielle (ext2, ext3, reiserfs, xfs) des disques et du système de fichiers.

Le serveur accède directement à cette image de système de fichiers; il s'agit de son propre système racine de fichiers. Les noeuds quant à eux y accèdent via le réseau par le protocole NFS. Pour éviter toute chance de corruption de l'image du système, seul le serveur a accès en écriture à cette image; les noeuds n'y ont accès qu'en lecture seulement.

Comme les noeuds ne possèdent pas de disques locaux, un mécanisme de démarrage à distance via réseau est employé. Le noyau est ainsi transféré à partir du serveur en direction des noeuds où il est ensuite exécuté localement. Le noyau est configuré pour utiliser le système de fichier du serveur exporté par NFS comme système de fichier racine.

Puisque le système racine de fichiers n'est accessible qu'en lecture seule à partir des noeuds, un mécanisme doit être mis en place pour permettre l'écriture de certains fichiers localement sur les noeuds afin de pouvoir procéder à une initialisation correcte du système. Pour ce faire, des copies locales de certains répertoires sont créées en mémoire sur les noeuds et viennent supplanter leurs pendants disponibles uniquement en lecture.

Deux mécanismes différents peuvent être utilisés pour ce faire. La méthode classique d'Adelie/SSI utilise les liaisons pour remplacer les repertoires originaux. Une seconde méthode développée utilise un mécanisme de superposition des répertoires, rendant le contenu initial toujours visible mais permettant d'effectuer des modifications à ce contenu dans une couche accessible en lecture et en écriture.

Comme certains fichiers pourraient nuire au bon fonctionnement des noeuds, un mécanisme permet l'exclusion de ces fichiers. D'autre fichiers doivent être quant à

eux uniques sur chaque noeud. Dans ce cas, un autre mécanisme est utilisé pour permettre de renommer ces fichiers en fonction d'un noeud propre ou d'un type de noeud.

Une fois ces changements mis en place, l'initialisation peut se poursuivre normalement.

Adelie/SSI met donc en place différents processus pour permettre le démarrage à distance des noeuds ainsi que le partage de l'image de système de fichiers du serveur. Pour éviter la corruption, les noeuds n'ont donc pas accès en écriture à cette image. Pour pallier à ce problème, un mécanisme est mis en place pour permettre aux noeuds d'avoir un accès en écriture à certains répertoires.

4.2.2 Démarrage réseau

Le processus de démarrage classique des noeuds sous Adelie/SSI s'effectue via le réseau. À l'aide de différents protocoles, le noeud obtient sa configuration, transfère l'image du noyau et du système de fichiers initial et importe le système de fichiers du serveur.

Bien qu'il soit possible d'utiliser Adelie/SSI via un mécanisme de démarrage traditionnel, c'est-à-dire à partir d'un noyeau sur disque, disquette, cdrom ou encore clé USB, il est plus simple de déployer et de maintenir à jour le noyau si l'on utilise un mécanisme de démarrage par réseau. De cette façon, une simple mise à jour du noyau sur le serveur sera refletée sur les noeuds lors du prochain démarrage.

Le mécanisme préconisé dans ce cas est un démarrage initial via PXE (INTEL CORPORATION, 1999). Il s'agit d'une norme introduite par Intel qui définit un mécanisme permettant la configuration d'un adaptateur réseau ainsi que le transfert

et l'exécution d'application. La norme PXE fait usage de divers protocoles déjà existants tels que ARP (PLUMMER, 1982), BOOTP (CROFT and GILMORE, 1985) (REYNOLDS, 1993) et TFTP (FINLAYSON, 1984) (SOLLINS, 1992) (MALKIN and HARKIN, 1998a) (MALKIN and HARKIN, 1998b) et spécifie une interface de fonctions additionnelles.

Un adaptateur réseau répondant à la norme PXE possède un progiciel *(firmware)*, ou client PXE, qui peut être activé au démarrage de la machine. Ce client, lorsqu'exécuté, va tenter d'obtenir une adresse IP ainsi que des informations supplémentaires, tel que le nom et le chemin d'accès du fichier à télécharger ainsi que l'adresse IP du serveur sur lequel se trouve le fichier en émettant une requête BOOTP.

Le fichier est ensuite transféré par TFTP puis exécuté localement. Typiquement, cet exécutable n'est qu'un client plus flexible pour ensuite télécharger un noyau ainsi qu'une image initiale. Dans le cas d'Adelie/SSI, PXELinux est utilisé. Il s'agit d'une composante de SysLinux permettant le démarrage réseau de divers systèmes d'exploitation. Il est donc possible à l'aide de PXELinux de spécifier un noyau ainsi que des paramètres additionnels, tel qu'une image de système initiale.

La figure 4.3 montre les principales interactions entre un noeud et le serveur lors du démarrage. Voici en détails les étapes de communications suite au démarrage et à l'initialisation matérielle du noeud et à l'exécution du progiciel de l'adaptateur réseau.

Le noeud diffuse *(broadcast)* une requête BOOTP de type BOOTREQUEST. Le serveur DHCP (DROMS, 1993) qui examine l'adresse MAC de la source et si elle correspond à une entrée dans son fichier de configuration, répond à cette dernière par une réponse BOOTP de type BOOTREPLY. Cette réponse contient l'adresse

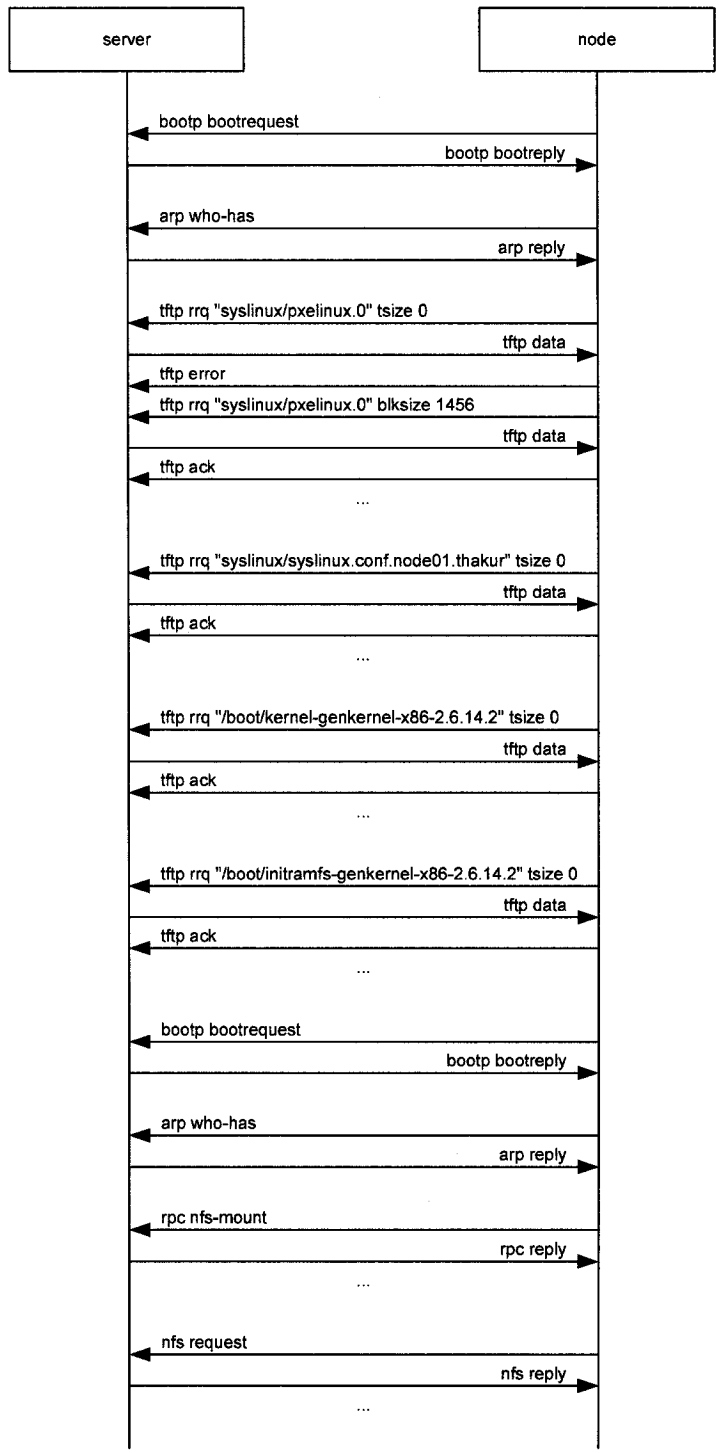

Figure 4.3 Communications entre le serveur et le noeud client lors du démarrage.

IP du noeud et celle du serveur TFTP ainsi que le nom et le chemin d'accès du code à télécharger. Des extensions permettent de spécifier des paramètres additionnels pour PXELinux, soit la racine du chemin d'accès ainsi que le nom du fichier de configuration spécifique au noeud.

Puisque l'adresse du serveur TFTP, fournie par le serveur DHCP, est une adresse IP, le noeud diffuse une requête ARP de type WHO-HAS. Le serveur, dont l'adresse IP correspond à celle de la requête ARP WHO-HAS répond par le biais d'une réponse ARP de type REPLY et indique ainsi son adresse MAC au noeud.

Le noeud envoie une requête TFTP de type RRQ. Cette requête contient le nom ainsi que le chemin d'accès du fichier à télécharger, dans ce cas-ci `syslinux/pxelinux.0`, soit le client PXELinux, ainsi que l'option TSIZE 0. Le premier paquet retourné par le serveur sera un paquet TFTP de type DATA, contenant la taille du fichier à télécharger. Si le noeud répondait par une réponse TFTP de type ACK, le serveur poursuivrait sa transmission en envoyant le fichier par paquet de 512 octets. Dans ce cas-ci, le noeud envoie une réponse TFTP de type ERROR afin de mettre fin à la transmission du fichier. Le noeud possède donc la taille du fichier mais ne l'a pas encore téléchargé.

Une seconde requête TFTP de type RRQ est envoyée au serveur. Celle-ci contient l'option BLKSIZE 1456. Cette option permet de spécifier en octets la taille des paquets à transmettre par le serveur. Il est ainsi possible d'effectuer le téléchargement du fichier en un nombre plus petit de paquets. Les performances en sont ainsi accrues. Le serveur répond par un envoi TFTP de type DATA contenant la taille du fichier auquel le noeud répond par une réponse TFTP de type ACK. L'échange se poursuit et le serveur transfère ainsi le fichier en multiples paquets, chaque paquet étant accepté par une réponse TFTP de type ACK par le noeud.

Le fichier téléchargé par le noeud est ensuite exécuté. Le client PXELinux intilialisé tente d'obtenir un fichier de configuration, dont le nom et le chemin d'accès ont été obtenus lors de la requête BOOTP initiale. Il envoie donc une requête TFTP de type RRQ pour télécharger son fichier de configuration. Le serveur répond initialement par une réponse TFTP de type DATA contenant la taille du fichier. Suite à une réponse TFTP de type ACK du noeud, le fichier est transféré en multiples blocs.

Ce fichier, similaire à la plupart des chargeurs pour Linux tel que *GRUB* ou *LILO*, spécifie le nom et le chemin d'accès du noyau ainsi que de l'image initiale de système . De plus, les paramètres optionnels du noyau y sont spécifiés. Dans le cas d'Adelie/SSI, il existe un fichier de configuration par noeud, ce qui permet d'avoir des configurations uniques si besoin est. Les paramètres optionnels de noyau utilisés permettent de spécifier l'utilisation du système de fichier racine via NFS (Network File system) (SUN MICROSYSTEMS, 1989) (CALLAGHAN et al., 1995) ainsi que le niveau de scripts d'initialisations à utiliser. Cela permet d'avoir différentes configations logicielles au sein d'une même grappe de calcul.

Par la suite, des requêtes TFTP RRQ sont faites pour télécharger le noyau, dans le cas présent `/boot/kernel-genkernel-x86-2.6.14.2`, suivi de l'image initiale de système , `/boot/initramfs-genkernel-x86-2.6.14.2`. Une fois ces deux fichiers obtenus, le noyau peut alors être lancé. Le système de fichier racine sera le même que celui du serveur, bien qu'exporté en lecture seule afin d'empêcher toute corruption possible des données du serveur.

Le noyau et l'image initiale de système sont alors décompressés et le noyau est exécuté. Il est important que le noyau ait été compilé avec les options `CONFIG_IP_PNP`, `CONFIG_IP_PNP_BOOTP`, `CONFIG_NFS_FS`, `CONFIG_NFS_V3`, `CONFIG_ROOT_NFS` ainsi que les pilotes pour les adaptateurs réseau présents sur les noeuds. Ces composantes

du noyau permettent l'autoconfiguration du réseau par le protocole BOOTP et permettent d'utiliser un système de fichier racine monté par NFS.

Lors de l'initialisation du noyau, les paramètres obtenus initialement du serveur DHCP ne sont pas conservés. Une seconde requête BOOTP de type BOOTREQUEST est alors émise lors de l'autoconfiguration de l'adaptateur réseau. La séquence est alors la même que lors de cette même requête initiale, de même que pour la requête ARP de type WHO-HAS qui s'en suit.

Une fois le noyau initialisé, le système de fichier racine est monté via NFS. Le protocole NFS ne couvre pas la portion de montage du système de fichier, il ne couvre que les transactions elles-mêmes. Le protocole de montage *(Mount Procolol)* est utilisé pour ce faire. Similaire au protocole NFS, il se compose d'une ensemble de procédures RPC.

Le noeud émet donc des requêtes RPC afin de s'identifier au serveur et de demander la permission de monter le répertoire racine pour le rendre accessible par NFS. Chaque requête est suivie d'une réponse et lorsque ces transactions sont terminées, le répertoire racine du serveur est disponible en lecture via NFS. C'est à ce point que le noyau lance l'exécution du processus d'initialisation se trouvant dans l'image initiale système.

Les protocoles ARP, BOOTP, TFTP et NFS sont utilisés pour permettre le démarrage à distance des noeuds. Bien qu'il soit possible d'utiliser d'autres mécanismes, celui-ci à l'avantage de ne nécessiter d'aucun support matériel pour contenir l'image du noyau, si ce n'est un adaptateur réseau répondant à la norme PXE. De plus, la mise à jour du système s'en voit ainsi simplifiée.

4.2.3 Initialisation de l'environnement des noeuds

Une fois l'initialisation du noyau terminée, celle du système prend la relève. C'est à ce point que doivent s'effectuer les étapes cruciales qui permettent aux noeuds de fonctionner malgré un accès en lecture simple au système de fichier racine. Pour ce faire, un espace local accessible en écriture doit être substitué aux répertoires nécessaires au bon fonctionnement du système. De cet espace, certains fichiers sont exclus et certains autres liés symboliquement pour permettre la différentiation des noeuds.

Après l'initialisation du noyau, le système exécute une image de système initiale compressée. Ce système est habituellement utilisé pour mettre en place l'environnement qui lui suivra. Sous Gentoo Linux, cette image est employée pour la détection et le chargement des pilotes qui pourraient être nécessaires pour le chargement du système de fichier racine. Les pilotes USB, SATA, RAID et autres sont donc chargés pour permettre d'accéder à un système de fichiers racine se trouvant sur un de ces périphériques.

Bien qu'Adelie/SSI n'utilise pas de support physique pour son fonctionnement, il est possible d'exploiter des disques locaux ou encore des clés USB sur les noeuds. C'est donc pour cette raison que l'image de système initiale, bien qu'accessoire au fonctionnement d'Adelie/SSI, est généralement utilisée.

Le niveau d'initialisation à parcourir est déterminé par le paramètre du noyau `softlevel=<host-type>`. Ainsi les noeuds qui doivent avoir un processus d'initialisation différent du serveur de par leur mode de fonctionnement se voient démarrer un ensemble de services différents de ceux du serveur. Il est même possible d'avoir différents types de noeuds, ce qui permet différentes fonctions : noeud frontal, noeud de visualisation, noeud de compilation, serveur web, proxy

ou n'importe quel autre rôle que l'on veut leur faire jouer.

Les noeuds suivent donc un processus d'initialisation différent du serveur. Par défaut, les niveaux d'initialisation classiques sont `boot` suivi de `default`. Dans le cas d'un noeud, il s'agit typiquement de `boot.node` suivi de `node`. Dans les deux cas, il est possible d'utiliser d'autres niveaux.

Pour que le reste du processus d'initialisation puisse s'exécuter correctement, le script d'initialisation d'Adelie/SSI, nommé `adelie-ssi`, doit être le premier à être exécuté sur les noeuds, et ce en lieu et place du script d'initialisation typique `checkroot`.

Lors de l'initialisation de l'environnement des noeuds, la configuration centrale ainsi que la configuration d'environnement est lue. Par la suite, un système de fichier local en mémoire vive est créé. Selon le mode de fonctionnement spécifié dans le fichier `/etc/conf.d/adelie-ssi`, le mécanisme classique par liaison ou le mécanisme expérimental par superposition est alors initié.

Dans le cas du mécanisme par liaison, les répertoires mentionnés à la rubrique `LOCAL_DIRS` des fichiers d'environnement seront ensuite copiés vers le répertoire en mémoire vive. Il s'agit typiquement des répertoires `/etc`, `/tmp` et `/var`, bien qu'il est possible à l'utilisateur de spécifier ceux qui lui sont nécessaires. De cette copie seront exclus tous les fichiers et répertoires mentionnés à la rubrique `EXCLUDED_FILES`. La figure 4.4 offre une vue des systèmes de fichiers sur le serveur ainsi que sur les noeuds.

Dans le cas du mécanisme par superposition, le système de fichiers par superposition unionfs est utilisé. Unionfs permet de superposer de façon hiérarchique un nombre arbitraitre de répertoires. Il en résulte une vue combinée des différentes couches, comme si celles-ci n'en formaient qu'une seule.

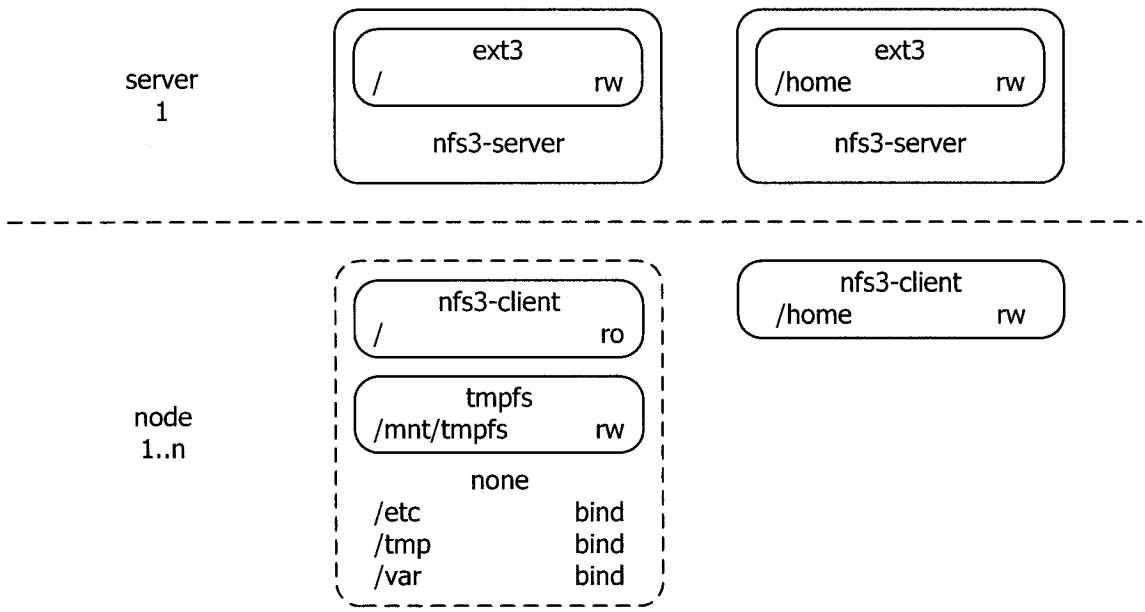

Figure 4.4 Détails des couches de systèmes de fichiers.

Il est ainsi possible de superposer une couche en lecture-écriture à une couche en lecture seule. Ainsi, les lectures pourront s'effectuer dans la couche en lecture seule, tandis que les écritures dans la couche en lecture-écriture. Unionfs utilise le principe d'écriture-sur-modification, ce qui veut dire que si une modification ne peut être enregistrée parce que le fichier appartient à une couche en lecture seule, une copie du fichier sera créée sur la couche en lecture-écriture.

Un mécanisme similiaire permet d'effacer des fichiers d'une couche en lecture seule, un fichier d'écrasement est alors créé pour indiquer au système de fichier de ne pas tenter de parcourir les couches inférieures pour tenter de trouver le fichier.

Donc, dans le cas du mécanisme par superposition, les répertoires mentionnés à la rubrique `LOCAL_DIRS` des fichiers d'environnement seront créés dans le système de fichier en mémoire vive, et ce sans contenu initial. Ces répertoires locaux seront ensuite superposés à l'aide de système de fichier unionfs à leur pendant du système de fichier racine en lecture seule. Des fichiers d'écrasement seront par la suite

générés pour effacer les fichiers apparaissant à la rubrique `EXCLUDED_FILES`.

Par la suite, l'initialisation se poursuit de la même manière peu importe le mécanisme employé.

Des liens symboliques vers certains fichiers de configuration qui sont spécifiques à chaque noeud ou à chaque type de noeud. Ainsi, le fichier `/etc/conf.d/hostname` qui doit être unique pour chaque noeud se voit lié symboliquement vers `/etc/conf.d/hostname.node01.thakur` par exemple. Ces fichiers sont spécifiés à la rubrique `LINKED_FILE_BY_FQDN` des fichiers d'environnement.

Dans d'autres cas, la configuration ne doit pas être unique à chaque noeud mais bien à chaque catégorie de noeud. C'est l'exemple de `/etc/syslog-ng/syslog-ng.conf`. La configuration est différente entre le serveur et les noeuds mais elle est la même pour tous les noeuds, donc tous les noeuds verront donc ce nom pointer vers `/etc/syslog-ng/syslog-ng.conf.node`. Ces fichiers sont quant à eux spécifiés à la rubrique `LINKED_FILES_BY_TYPE`.

Ce mécanisme permet ainsi d'avoir des configurations différentes sans avoir à modifier les scripts d'initialisations des différents services. Cela rend donc transparente l'utilisation d'Adelie/SSI.

Enfin, quelques modifications locales additionnelles sont effectuées. La table des systèmes de fichiers montés, `/etc/mtab` est regénérée car l'originale est celle du serveur. La partition racine est remontée, toujours en lecture seule, pour tenir compte des paramètres différents qui pourraient se trouver dans le fichier de définition des systèmes de fichiers, `/etc/fstab`. Finalement, le nom du noeud est corrigé dans le profil du système.

4.2.4 Détection et configuration des noeuds

La mise en place de l'environnement d'une grappe de calcul est un processus long qui nécessite une connaisance du fonctionnement de bon nombre de services. Afin de permettre à l'administrateur non-initié de déployer et de configurer rapidement un environnement complet, un mécanisme de configuaration automatique des services requis a été intégré à Adelie/SSI.

Ce mécanisme se compose d'un ensemble d'outils pour gérér l'intégration de nouveaux noeuds ainsi que la configuration de l'ensemble des noeuds. Il existe donc un outil responsable de la détection et de l'intégration des nouveaux noeuds au sein de la configuration centrale. De plus, un outil de configuration est mis à disposition pour centraliser l'utilisation des modules de configuration.

Sous Adelie/SSI, la configuration des services se définit par le fichier de configuration centralisé. Il s'agit d'un fichier principal, `/etc/adelie/adelie.conf`, qui peut faire appel par inclusion à des sous-fichiers de même type. Ces fichiers contiennent la configuration globale de la grappe de calcul. C'est ici que l'on spécifie l'ensemble des paramètres qui seront utilisés plus tard par les modules de configuration pour regénérer les fichiers de configuration spécifiques à chaque service.

Ces fichiers comportent une structure similaire au language C en certains aspects. Le format a été emprunté au modèle de fichier de configuration du logiciel ISC DHCP. Il y a deux types de structures possibles, soit les énoncés simples et les blocs.

Les énoncés simples sont composés d'un mot clé suivi d'un ou de deux paramètres et se terminent par un point-virgule. Ces énoncés affectent la configuration des noeuds qui les suivent jusqu'à la fin du bloc courant. Les énoncés n'ayant qu'un paramètre définissent des propriétés des noeuds. Les énoncés ayant deux paramètres définis-

sent quant à eux des propriétés des adaptateurs réseaux des noeuds.

Les blocs sont similaires aux énoncés simples. Il sont composés de la même forme, c'est-à-dire un mot clé suivi de un ou deux paramètres. Une accolade ouvrante qui suit vient définir l'ouverture d'un bloc logique. Le mot clé du bloc s'applique au contenu de ce bloc et ce jusqu'à la rencontre de l'accolade fermante correspondante. Il est possible de définir des sous-blocs à l'intérieur des blocs.

Une seule instruction ne s'applique pas au bloc, il s'agit du mot clé `include`. De plus, la définition d'un bloc prend un sens particulier avec le mot clé `host-name`. En effet, lors de la fermeture d'un bloc de type `host-name`, la configuration telle que définie immédiatement avant la fermeture de l'accolade est utilisée comme configuration pour le noeud spécifié par ce bloc. Donc les noeuds doivent impérativement être définis par un bloc et non par un énoncé simple.

Voici la liste des mots clés supportés :

- `cluster-name <name>` : Nom de la grappe de calcul. Habituellement le même que le nom d'hôte du serveur.
- `default-gateway <ip-address>` : Passerelle principale. Pour les noeuds il s'agit habituellement de l'adresse du serveur. Dans le cas du serveur, il s'agit de la passerelle principale pour rejoindre l'extérieur.
- `dns-domain-name <interface> <name>` : Nom de domaine de l'interface. Pour l'interface principale, il s'agit habituellement du nom d'hôte du serveur. Pour les autres interfaces, on précède habituellement par le nom du type de l'interface.
- `home-path <path>` : Chemin d'accès du répertoire maison sur le serveur. Il s'agit habituellement de `/home`.

- `home-server <ip-address>` : Adresse du serveur où se situe le répertoire maison. Habituellement, il s'agit du serveur.

- `host-name <name>` : Forme courte du nom d'hôte. Habituellement le même que le nom de la grappe pour le serveur et nodeXX pour les noeuds.

- `host-type <softlevel>` : Type de noeud. Habituellement server pour le serveur et node pour les noeuds.

- `include <file>` : Nom du fichier à inclure. Il s'agit habituellement de fichiers se trouvant dans le répertoire `/etc/adelie`.

- `ip-address <interface> <ip-address>` : Adresse de l'interface réseau. Les adresses de type privées débutent habituellement par 192.168.X.1 le serveur et se suivent pour les noeuds.

- `mac-address <interface> <mac-address>` : Adresse matérielle de l'interface réseau. Unique à chaque adaptateur réseau.

- `network-mask <interface> <network-mask>` : Masque de l'interface réseau. Varie selon la taille de la grappe de calcul, habituellement 255.255.255.0.

- `nis-domain-name <name>` : Nom du domaine d'indentification. Habituellement le même que le nom d'hôte du serveur.

- `nis-server <ip-address>` : Adresse du serveur d'identification. Habituellement, il s'agit du serveur.

- `primary-interface <interface>` : Interface réseau principale. Il s'agit habituellement de l'interface utilisée par les noeuds pour leur démarrage réseau. Sur le serveur, il s'agit de l'interface utilisée par le serveur DHCP.

- `processor-count <count>` : Nombre de processeurs. Il s'agit du nombre de processeurs qui seront mis à la disposition du système de traitement en lot. Habituellement 0 pour le serveur.
- `processor-type <architecture>` : Architecture du processeur. `ppc`, `x86` où `x86_64` selon le type de processeurs utilisés.
- `root-path <path>` : Chemin d'accès du répertoire racine sur le serveur. Il s'agit habituellement de `/`.
- `root-server <ip-address>` : Adresse du serveur où se situe le répertoire racine. Habituellement, il s'agit de l'adresse ip du serveur.

Un outil permet de faciliter l'ajout de noeuds supplémentaires. Une fois l'entrée de la configuration de base effectuée, cet outil détecte la présence de nouveaux noeuds sur le réseau. Ces noeuds peuvent ensuite être ignorés ou ajoutés à un fichier de configuration particulier.

Cet outil utilise le logiciel tcpdump pour surveiller la présence de requête de type bootp sur l'interface principale du serveur. Ces requêtes sont émises lors du démarrage d'un environnement de type PXE. Pour s'assurer qu'il ne s'agit pas d'une requête provenant d'un noeud déjà configuré ou encore d'un poste de travail se trouvant sur le même sous-réseau, une liste d'adresses matérielles à exclure est générée.

Lorsqu'une nouvelle adresse est alors détectée, cette adresse est placée dans une liste de noeuds en attente de configuration. Il s'agit du fichier `/etc/adelie/adelie-detect.new` qui contient la liste des adresses matérielles détectées mais non configurée.

La configuration peut s'effectuer ultérieurement, au moment jugé opportun par

l'utilisateur, ou immédiatement après la détection de chaque nouveau noeud. Lors de la configuration, il est possible d'exclure une adresse matérielle si, par exemple, elle appartient à un poste de travail se trouvant sur le même sous-réseau et qui n'est pas sous le contrôle d'Adelie/SSI. Dans ce cas, l'adresse matérielle est alors placée dans le fichier `/etc/adelie/adelie-detect.deny`.

Si l'adresse détectée n'est pas à ignorer, elle peut alors être configurée. Deux paramêtres sont alors configurables : le nom d'hôte du noeud ainsi que son adresse IP. Le logiciel tente de déduire la bonne configuration à proposer à l'utilisateur pour ces paramètres. Pour cela, il utilise la configuration déjà existante. Il arrive ainsi à déduire la dernière adresse IP disponible, ainsi que les éléments qui permettent de générer le nom d'hôte.

Un nom d'hôte de noeud sous Adelie/SSI est typiquement constitué d'un nom de base suivi d'un indice qui peut être précédé de zéro pour aligner les nombres. Le logiciel va donc tenter de déduire le nom de base, le dernier indice, ainsi que le nombre total de zéros à utiliser pour l'alignement. Dans le cas d'une ambiguité, il est possible de spécifier ces paramètres lors de l'invocation.

Toute cette configuration peut se dérouler automatiquement ou de manière interactive, dans lequel cas des suggestions seront proposées à l'utilisateur. Une fois la configuration acceptée, elle se verra ajoutée au fichier de configuration `/etc/adelie/adelie-detect.conf`. Ce fichier se trouve typiquement inclu par le fichier `/etc/adelie/adelie.conf` dans la section contenant la configuration des noeuds. S'il s'agit d'une grappe homogène, il n'y a habituellement pas de configuration additionnelle à apporter.

Une fois que la configuration centrale est entrée et que les noeuds ont été ajoutés manuellement où par détection, les modules de configuration sont utilisés pour

regénérer les fichiers de configuration des divers services à partir des informations se trouvant dans la configuration centrale.

Les modules de configurations sont situés dans le répertoire `/usr/share/adelie/-conf.d`. Bien qu'il soit possible de les invoquer directement, un utilitaire, *adelie-configure*, permet d'invoquer un ensemble de modules tout en résolvant les dépendances entre ceux-ci. Il suffit de spécifier la liste des noms de services à reconfigurer, ou `all` pour tous.

Les modules de configuration utilisent donc la configuration centrale de la grappe pour déterminer les paramètres à spécifier dans les fichiers de configuration des différents services. De plus, chaque module peut se voir associé un fichier d'environnement se trouvant dans le répertoire `/usr/share/adelie/env.d`.

Ce fichier qui est également utilisé par le processus d'initialisation, permet de spécifier si les fichiers de configuration générés devront être uniques pour chaque noeud ou encore pour chaque groupe de noeuds. Pour ce faire, le nom du fichier à générer est comparé avec la liste des fichiers uniques par noeud, `LINKED_FILES_BY_FQDN`, ou encore uniques par type de noeud, `LINKED_FILES_BY_TYPE`.

Une fois les fichiers de configuration regénérés, il se retrouveront automatiquement sur les noeuds si l'on utilise la méthode expérimentale par superposition. Dans le cas échéant, il faut les y transférer ou encore redémarrer les noeuds pour resynchroniser le tout.

Les modules de configuration sont donc composés d'un ensemble d'heuristiques écrites en script Bash, faisant appel à une librairie de fonctions offertes par Adelie/SSI, qui simplifie ainsi leur création. Ces heuristiques sont la distillation du savoir et de l'expérience de configuration et d'administration de grappes de calcul.

En définissant ainsi un format simple de configuration centrale, l'utilisateur peut configurer avec une granularité intéressante une grappe de calcul ou un réseau de postes de travail partageant une image unique de système de fichiers. À l'aide d'outils de détection et de configuration, il est alors possible d'intégrer aisément de nouveaux noeuds et de reconfigurer l'ensemble des services du système. Tous ces outils permettent donc à un utilisateur novice de déployer et de gérer facilement une configuration robuste et éprouvée.

4.3 Services

La mise en place d'une grappe de calcul ressemble beaucoup à l'assemblage d'un puzzle. Un grand nombre de services est à la disposition de l'administrateur qui a alors pour tâche de choisir les bons, de les réunir et de les configurer. L'utilisation d'Adelie/SSI permet de simplifier le tout en réunissant la majorité des services nécessaires.

Bien que ces services soient installés et configurés automatiquement, il est tout de même important de connaître l'impact de l'installation et de l'utilisation d'Adelie/SSI sur l'ensemble de ces services. Une description de ces services ainsi que des fichiers pris en charge sous Adelie/SSI est donc nécessaire.

4.3.1 adelie-ssi

Il s'agit du service fondamental d'Adelie/SSI qui a pour rôle de gérer la mise en place de l'environnement particulier des noeuds.

Aucun service n'est utilisé sur le serveur. Le service utilisé sur les noeuds est `adelie-ssi`.

Les fichiers de configurations modifiés lors de l'installation sont `/etc/conf.d/adelie-ssi` et `/etc/conf.d/hostname`. Les fichiers de configurations modifiés lors de la configuration sont `/etc/conf.d/hostname`, `/etc/hosts`, `/etc/conf.d/modules` et `/etc/conf.d/net`.

4.3.2 adelie-ssi-config-dhcp

Le service DHCP est offert par le serveur et permet de spécifier la configuration réseau des noeuds. Il entre également en jeu dans le processus de démarrage réseau.

Le service utilisé sur le serveur est `dhcp`. Aucun service n'est utilisé sur les noeuds.

Les fichiers modifiés lors de l'installation sont `/etc/conf.d/dhcp` et `/etc/dhcp/dhcpd.conf`. Les fichiers modifiés lors de la configuration sont `/etc/conf.d/dhcp` et `/etc/dhcp/dhcpd.conf`.

4.3.3 adelie-ssi-config-ganglia

Ganglia permet d'avoir une vue graphique de l'état complet de la grappe et ce via des pages web. Il s'avère un outil très utile pour diagnostiquer les problèmes de fonctionnement. La figure 4.5 présente une des vues disponibles.

Les services utilisés sur le serveur sont `apache2`, `gmetad` et `gmond`. Le service utilisé sur les noeuds est `gmond`.

Les fichiers modifiés lors de l'installation sont `/etc/conf.d/apache2`, `/etc/init.d/gmetad`, `/etc/init.d/gmond`, `/etc/gmetad.conf` et `/etc/gmond-global.conf`. Les fichiers modifiés lors de la configuration sont `/etc/gmetad.conf` et `/etc/gmond.conf`.

4.3.4 adelie-ssi-config-nfs

Le protocole NFS est utilisé entre autre pour exporter le système de fichiers racine vers les noeuds.

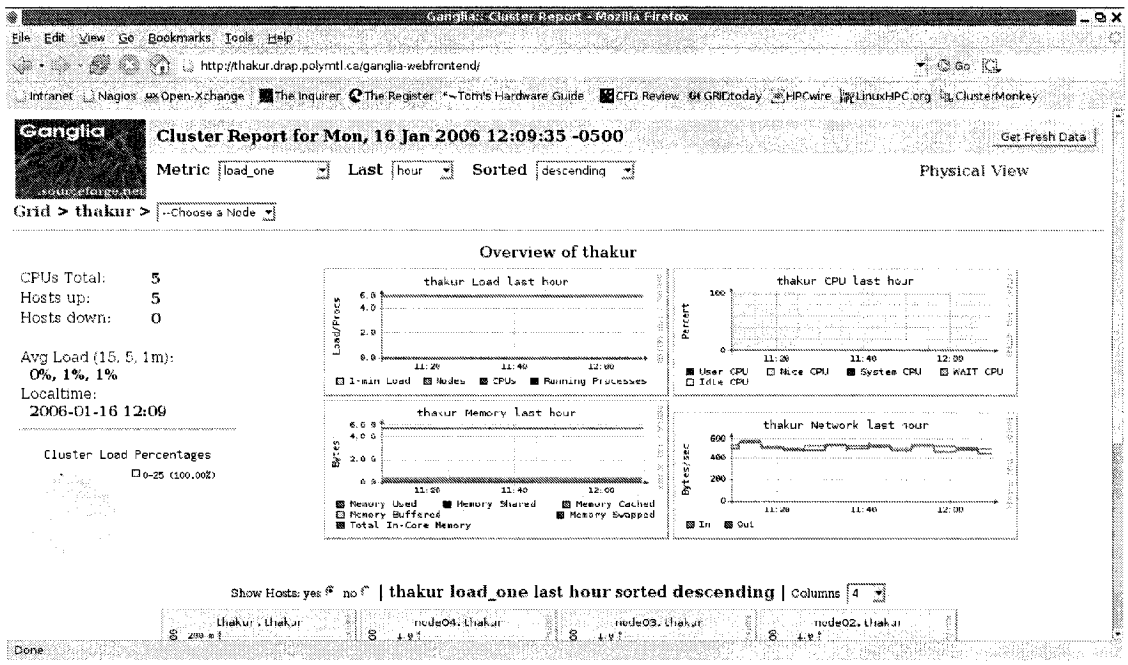

Figure 4.5 Système de surveillance Ganglia.

Les services utilisés sur le serveur sont `nfs` et `nfsmount`. Le service utilisé sur les noeuds est `nfsmount`.

Il n'y a aucun fichier modifié lors de l'installation. Les fichiers modifiés lors de la configuration sont `/etc/exports` et `/etc/fstab`.

4.3.5 adelie-ssi-config-nis

Le service NIS permet la centralisation de l'authentification au travers de la grappe. Il est ainsi possible d'éviter les problèmes de synchronisation des listes d'utilisateurs.

Les services utilisés sur le serveur sont `rpc.yppasswdd`, `ypbind` et `ypserv`. Le service utilisé sur les noeuds est `ypbind`.

Les fichiers modifiés lors de l'installation sont `/etc/publickey` et `/var/yp/se-`

curenets. Les fichiers modifiés lors de la configuration sont /etc/-conf.d/domainname, /etc/group, /etc/gshadow, /etc/passwd, /etc/shadow, /etc/yp.conf et /var/yp/securenets.

4.3.6 adelie-ssi-config-ntp

Le protocole NTP est utilisé pour synchroniser les horloges des noeuds à celle du serveur.

Le service utilisé sur le serveur est ntpd. Le service utilisé sur les noeuds est ntpd.

Le fichier modifié lors de l'installation est /etc/ntp.conf. Le fichier modifié lors de la configuration est /etc/ntp.conf.

4.3.7 adelie-ssi-config-openib

La pile OpenIB est responsable de l'intégration de matériel de communication Infiniband au sein de la grappe.

Le services utilisés sur le serveur sont net.ib0 et opensm. Les services utilisés sur les noeuds sont net.ib0 et opensm.

Le fichier modifié lors de l'installation est /etc/dat.conf. Le fichier modifié lors de la configuration est /dat/ntp.conf.

4.3.8 adelie-ssi-config-openssh

OpenSSh est utilisé pour permettre la connexion entre les noeuds de la grappe. Un mécanisme d'authentification d'hôte par clé est employé pour permettre une

connexion sans mot de passe au sein de la grappe.

Le service utilisé sur le serveur est `sshd`. Le service utilisé sur les noeuds est `sshd`.

Les fichiers modifiés lors de l'installation sont `/etc/conf.d/sshd`, `/etc/init.d/sshd`, `/etc/ssh/ssh_config` et `/etc/ssh/sshd_config`.

Les fichiers modifiés lors de la configuration sont `/etc/hosts.equiv`, `/etc/ssh/ssh_known_hosts`, `/etc/ssh/ssh_host_rsa_key`, `/etc/ssh/ssh_host_rsa_key.pub`, `/home/root/.ssh/authorized_keys`, `/home-/root/.ssh/id_rsa` et `/home/root/.ssh/id_rsa.pub`.

4.3.9 adelie-ssi-config-syslinux

La composante pxelinux de syslinux est utilisée lors du processus de démarrage réseau des noeuds. Il s'agit d'un client pxe configurable qui permet de télécharger et d'exécuter le noyau.

Aucun service n'est utilisé sur le serveur. Aucun service n'est utilisé sur les noeuds.

Le fichier modifié lors de l'installation est `/boot/syslinux/pxelinux.0`. Le fichier modifié lors de la configuration est `/boot/syslinux/syslinux.conf`.

4.3.10 adelie-ssi-config-syslog-ng

Le logiciel Syslog-NG est utilisé pour gérer les journaux du système. Ces journaux sont redirigés des noeuds vers le serveur de manière à réduire l'utilisation d'espace mémoire requise par le répertoire `/var/log` sur les noeuds.

Le service utilisé sur le serveur est `syslog-ng`. Le service utilisé sur les noeuds est

`syslog-ng`.

Le fichier modifié lors de l'installation est `/etc/syslog-ng/syslog-ng.conf`. Le fichier modifié lors de la configuration est `/etc/syslog-ng/syslog-ng.conf`.

4.3.11 adelie-ssi-config-tftp-hpa

Le serveur TFTP-HPA est utilisé pour télécharger le client pxelinux ainsi que le noyau lors du démarrage réseau des noeuds.

Le service utilisé sur le serveur est `in.tftpd`. Aucun service n'est utilisé sur les noeuds.

Le fichier modifié lors de l'installation est `/etc/conf.d/in.tftpd`. Il n'y a aucun fichier modifié lors de la configuration.

4.3.12 adelie-ssi-config-torque

Torque, qui est une branche d'OpenPBS, est utilisé sur la grappe pour gérer les ressources de celle-ci. En particulier, il est utilisé pour gérer l'accès aux noeuds de manière exclusive.

Les services utilisés sur le serveur sont `pbssched`, `pbsserver`. Le service utilisé sur les noeuds est `pbsmom`.

Les fichiers modifiés lors de l'installation sont `/var/spool/pbs/sched_priv/sched_config`, `/var/spool/pbs/server_priv/qmgr_default` et `/var/spool/pbs/mom_priv/config`. Les fichiers modifiés lors de la configuration sont `/var/spool/pbs/mom_priv/config`, `/var/spool/pbs/server_priv/nodes`

et /var/spool/pbs/server_name.

4.4 Historique

Depuis sa conception initiale au sein du groupe de calcul de haute perfomance du centre de recherches en calcul appliqué, Adelie Linux a connu de nombreuses modifications. Afin de situer le projet dans le temps et d'aider à la transition entre les différentes versions, il est important de voir les points saillants de cette évolution.

4.4.1 Adelie Linux 1.0.0

2002. L'équipe de développement intiale comprend Sylvain Fourmanoit et Benoît Morin au sein du groupe de calcul de haute performance du centre de recherche en calcul appliqué.

Le nom d'hôte `server` est utilisé pour différencier le serveur des noeuds. Le niveau d'initialisation `boot` est commun au serveur et aux noeuds. Les niveaux d'initialisatons `default_server` et `default_node` sont ensuite utilisés respectivement par le serveur et les noeuds.

Les scripts de Gentoo Linux `/sbin/rc`, `/etc/init.d/functions.sh` et `/etc/init.d/halt.sh` sont modifiés pour permettre l'initialisation de l'environnement des noeuds.

Ajout des fichiers d'environnement `/etc/exclude/etc`, `/etc/exclude/tmp` et `/etc/exclude/var`. Ajout du script d'initialisation `/etc/init.d/switch`.

4.4.2 Adelie/SSI 1.0

Mai 2003. La portion image de système de fichiers unique d'Adelie Linux est renommée Adelie/SSI. Jean-François Richard et Olivier Crête se joignent au groupe.

Le paramètre du noyau `gentoo=adelie` est utilisé pour différencier le serveur des noeuds.

4.4.3 Adelie/SSI 2.0

Juillet 2003. Le développement est tranféré au laboratoire de recherche en applications parallèles de l'école Polytechnique de Montréal. Sylvain Fourmanoit quitte le groupe.

Le paramètre du noyau `softlevel=<host-type>` est utilisé pour différencier le serveur des noeuds et pour déterminer le type du noeud. Les niveaux d'initialisation `boot` et `default` sont utilisés par le serveur, `boot.<host-type>` et `<host-type>` sont utilisés par les noeuds.

Les fichiers d'environnement `/etc/exclude/etc`, `/etc/exclude/tmp` et `/etc/exclude/var` sont remplacés par le fichier `/etc/conf.d/adelie`.

Le script d'initialisation `/etc/init.d/swtich` est supprimé. Les modifications aux scripts de Gentoo Linux `/sbin/rc` et `/sbin/init.d/halt.sh` sont respectivement déplacées vers les scripts d'initialisation `/etc/init.d/adelie` et `/etc/init.d/adeliepost`.

4.4.4 Adelie/SSI 2.1

Août 2003. Les modifications au système Gentoo Linux sont intégrés dans la distribution standard.

Le script d'initialisation `/etc/init.d/adeliepost` est supprimé.

Ajout des fichiers de contrôle des dépendances `/etc/runlevels/boot.node/.critical` et `/etc/runlevels/boot.node/.fake`.

4.4.5 Adelie/SSI 2.2

Mars 2004. Oliver Crête quitte l'équipe.

Correction d'erreurs.

4.4.6 Adelie/SSI 2.3

Juillet 2004.

Le fichier d'environnement `/etc/conf.d/adelie` est renommé `/etc/adelie.d/adelie`. Ajout des fichiers d'environnement `/etc/adelie.d/dhcp`, `/etc/adelie.d/iptables`, `/etc/adelie.d/nfs`, `/etc/adelie.d/nis`, `/etc/adelie.d/syslog-ng` et `/etc/adelie.d/torque`.

Ajout des modules de configuration `/usr/sbin/passwd-update` et `/usr/sbin/ssh-update`.

4.4.7 Adelie/SSI 2.3.1

Août 2004.

Ajout du fichier de configuration `/etc/adelie.conf`. La structure est similaire à celle de /etc/hosts. Les paramètres suivants y sont représentés : `host-name` et `host-type`.

Ajout des fichiers d'environnement `/etc/adelie.d/ganglia` et `/etc/adelie.d/ntp`.

Ajout des outils `/usr/sbin/adelie-cluster-scp` et `/usr/sbin/adelie-cluster-ssh`.

Les modules de configuration `/usr/sbin/passwd-update` et `/usr/sbin/ssh-update` sont renommés respectivement `/usr/sbin/adelie-update-passwd` et `/usr/sbin/adelie-update-keys`.

4.4.8 Adelie/SSI 3.0.0

Septembre 2005.

Redéfinition du format de fichier de configuration. La structure est similaire à celle de `/etc/dhcp/dhcpd.conf`. Les mots clés suivants sont supportés : `cluster-name`, `dns-domain-name`, `host-name`, `host-type`, `ip-address`, `mac-address`, `nis-domain-name`, `primary-interface` et `processor-count`.

Le fichier de configuration `/etc/adelie.conf` est renommé `/etc/adelie/adelie.conf`.

Le fichier d'environnement /etc/adelie.d/iptables est supprimé. Les fichiers d'environnement /etc/adelie.d/adelie, /etc/adelie.d/dhcp, /etc/adelie.d/ganglia, /etc/adelie.d/nfs, /etc/adelie.d/nis, /etc/adelie.d/ntp, /etc/adelie.d/syslog-ng et /etc/adelie.d/torque sont renommés respectivement /usr/share/adelie/env.d/adelie-ssi.env, /usr/share/adelie/env.d/dhcp.env, /usr/share/adelie/env.d/ganglia.env, /usr/share/adelie/env.d/nfs.env, /usr/share/adelie/env.d/nis.env, /usr/share/adelie/env.d/ntp.conf, /usr/share/adelie/env.d/syslog-ng.env et /usr/share/adelie/env.d/torque.conf.

Le script d'initialisation /etc/init.d/adelie est renommé /etc/init.d/adelie-ssi.

Les outils /usr/sbin/adelie-cluster-scp et /usr/sbin/adelie-cluster-ssh sont renommés respectivement /usr/bin/adelie-copy et /usr/bin/adelie-execute. Ajout des outils /usr/bin/adelie-list, /usr/bin/adelie-ping et /usr/sbin/adelie-configure.

Les modules de configuration /usr/sbin/adelie-update-keys et /usr/sbin/adelie-update-passwd sont renommés respectivement /usr/share/adelie/conf.d/openssh.conf et /usr/share/adelie/conf.d/nis.conf. Ajout des modules de configuration /usr/share/adelie/conf.d/adelie-ssi.conf, /usr/share/adelie/conf.d/dhcp.conf, /usr/share/adelie/conf.d/ganglia.conf, /usr/share/adelie/conf.d/nfs.conf, /usr/share/adelie/conf.d/ntp.conf, /usr/share/adelie/conf.d/syslinux.conf, /usr/share/adelie/conf.d/syslog-ng.conf et /usr/share/adelie/conf.d/torque.conf.

4.4.9 Adelie/SSI 3.0.1

Octobre 2005.

Correction d'erreurs.

4.4.10 Adelie/SSI 3.1.0

Janvier 2006.

Ajout des mots clés `default-gateway`, `home-path`, `home-server`, `include`, `network-mask`, `nis-server`, `processor-type`, `root-path` et `root-server` au format de fichier de configuration.

Ajout de l'outil `/usr/sbin/adelie-detect`.

Ajout du module de configuration `/usr/share/adelie/conf.d/opeib.conf`.

CHAPITRE 5

ÉVALUATION DE PERFORMANCE

Différents aspects de performance sont d'un intérêt particulier dans ce contexte. Ces aspects sont utilisés pour juger de la viabilité des solutions retenues ainsi que pour jeter des pistes envisageables pour améliorer la solution déployée.

En premier lieu, la performance des interconnexions est évaluée. Il s'agit particulièrement de la performance entre les noeuds embarqués qui est importante car elle représente le goulot d'étranglement principal à l'expansion d'une solution à plusieurs puces.

Ensuite, les performances logicielles sont importantes car elles déterminent l'utilisabilité de la solution développée. En particulier, les performances liées au démarrage des noeuds ainsi qu'à la configuration de la grappe. De plus, des améliorations au système, tel que la mise en place d'un mécanisme de cache ou encore le développement d'une solution basée sur unionfs, sont évaluées pour juger de leur validité.

5.1 Matériel

Afin de développer des systèmes dont la complexité dépasse les limites en ressources d'une simple puce reprogrammable de type FPGA, la mise en grappe de plusieurs de celles-ci devient une solution intéressante. Par contre, pour que cette expansion des limites de complexité résulte en une solution acceptable, il est primordial que le mécanisme utilisé offre des performances intéressantes.

Dans le contexte de la grille de Virtex II Pro qu'est la couche embarquée de Thakur, la solution d'interconnexion retenue était basée sur la technologie gigabit ethernet. Chaque noeud est relié à ses voisins par un lien direct exclusif selon une topologie de grille. Chaque noeud possédait également deux processeurs indépendants.

Diverses modifications apportées au projet tout au long du développement de la carte ont privé celui-ci de la performance tant essentielle.

5.1.1 Bande passante TCP/IP

Afin d'obtenir une vision globale de la performance des interconnexions réseau, la bande passante est évaluée entre les différents noeuds de la grappe de calcul. Ainsi, tout les canaux de communications sont évalués à l'aide de l'application *NetPerf*.

La figure 5.1 présente les résultats d'un test de performance pour évaluer la bande passante au niveau de la couche TCP/IP en fonction de la taille des messages transmis.

Ces résultats ont été obtenus à l'aide de l'application d'étalonnage *NetPerf*. Un script particulier a été développé pour automatiser partiellement la collecte des résultats.

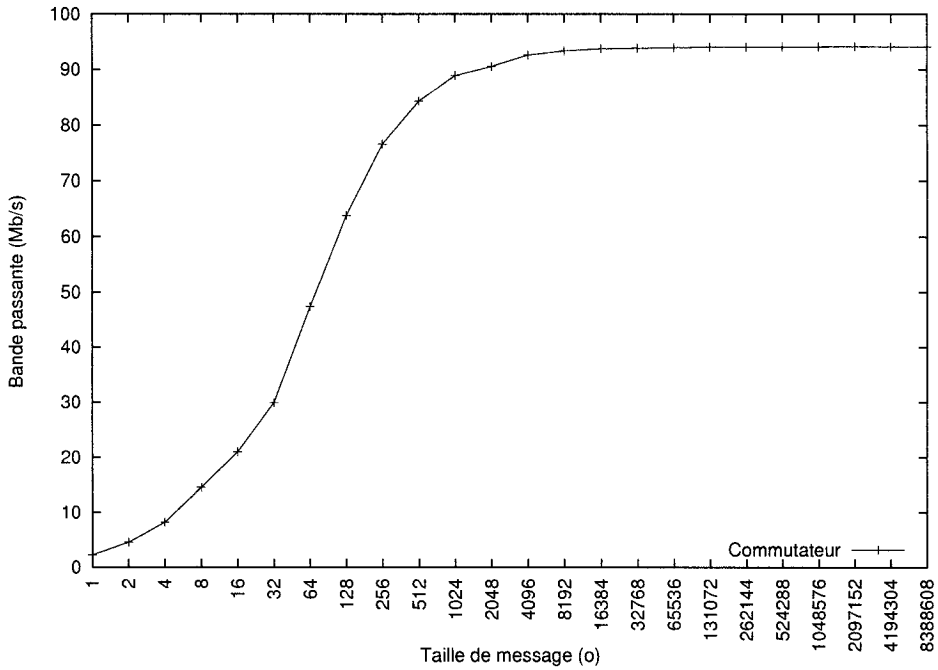

Figure 5.1 Évaluation de la performance TCP entre noeuds hôtes sur une grappe de calcul hétérogène.

Cette courbe représente la bande passante de la couche TCP/IP entre deux noeuds de type hôte reliés entre eux à travers un commutateur. Les liens utilisés sont du fast ethernet à 100 Mb/s, bien que le commutateur soit un commutateur gigabit ethernet.

Comme il est possible de le constater, la bande passante plafonne à près de 93 Mb/s par seconde, ce qui est réellement le maximum que l'on peut attendre de la technologie fast ethernet. De plus, ce maximum est atteint pour une taille de message d'environs 2048 octets. Cela correspond à la taille du MTU qui est de 1500 octets.

Donc cette courbe sert de référence pour comparer la performance des autres. La figure 5.2 montre la courbe de bande passante entre deux noeuds. Ici, la performance est évaluée lors d'un branchement direct et lors d'un branchement indirect

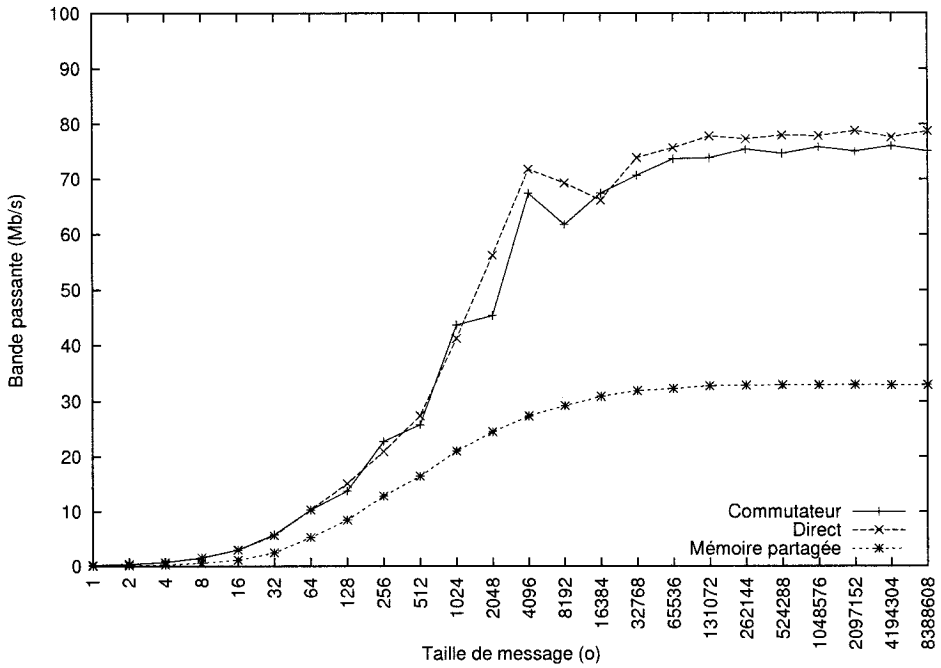

Figure 5.2 Évaluation de la performance TCP entre noeuds embarqués sur une grappe de calcul hétérogène.

à travers un commutateur. Dans les deux cas, les liens utilisés sont du gigabit ethernet à 1 Gb/s. De plus, la performance de la communication par zone de mémoire partagée est aussi présentée.

Les courbes de bande passante via gigabit ethernet sont éloquentes en ce qui a trait à la performance de celles-ci. La pointe de bande passante se situe juste en deçà du 80 Mb/s. La bande passante du gigabit ethernet devrait être près de dix fois supérieure. Cette performance est même en deçà de ce que l'on obtient pour du fast ethernet.

Différentes raisons peuvent expliquer en partie cette piètre performance. Les noeuds embarqués utilisent un noyau Linux 2.4, qui est réputé pour la lourdeur de sa pile TCP/IP. De plus, l'interface gigabit ethernet est relié au bus PLB qui a une cadence de 160 MHz et une largeur de 32 bits. Ce bus sature donc à 640 Mo/s. Le contrôleur

de mémoire ainsi que la seconde interface gigabit ethernet se trouvent également sur ce bus.

Bien que tous ces facteurs peuvent expliquer une performance sub optimale, l'évidence reste que le contrôleur gigabit ethernet ne fonctionne même pas à la hauteur d'un contrôleur fast ethernet.

Il en va de même pour le contrôleur en mémoire partagée. Celui-ci plafonne à environ 32 Mb/s et ce pour des messages de plus de 16 Ko. Il est donc plus performant de communiquer via ethernet entre deux processeurs se trouvant sur la même puce.

La figure 5.3 présente les performances de communication par ethernet entre un noeud hôte et un noeud embarqué. La communication doit absolument se faire au travers du commutateur car il est impossible de faire fonctionner l'interface fast ethernet du noeud hôte avec l'interface gigabit ethernet du noeud embarqué.

Il apparaît très clairement sur ces courbes la directionnalité du problème de communication. Les communications du noeud hôte vers le noeud embarqué plafonnent à près de 93 Mb/s, soit la même bande passante qu'entre les noeuds hôtes. Par contre, la bande passante du noeud embarqué vers le noeud hôte se limite à près de 65 Mb/s. Il est possible d'en conclure que la performance à l'émission est le facteur limitant de cette interface.

L'explication de la performance limitée du bus sur lequel est reliée cette interface couplé aux performances limitées du processeur qui lui est cadencé à 240 MHz semble valide.

Il a donc été démontré que les performances de communications entres les plateformes embarquées sont extrèmement limitées. Des choix de conceptions au niveau

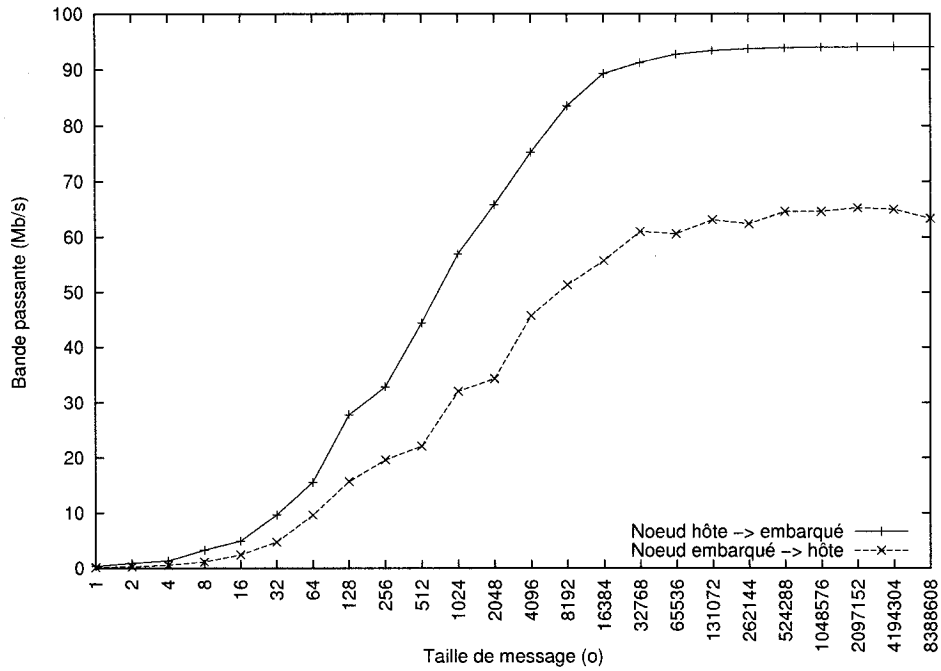

Figure 5.3 Évaluation de la performance TCP entre noeud hôte et embarqué sur une grappe de calcul hétérogène.

matériel ainsi qu'au niveau du progiciel de la plateforme limitent les potentiels d'interactions de celle-ci. De plus, des performances sub optimales au point de vue des liens gigabit ethernet n'offrent pas plus de voie de salut.

Une autre approche a été proposée par Amirix. Elle consiste à remplacer les liens gigabit ethernet par un autre mécanisme de communication plus léger. Ce changement implique la perte de connectivité vers une tierce architecture comme c'est le cas dans la grappe hétérogène Thakur. De plus, la couche physique de cette interconnexion limite à quelques centimètres la distance pouvant séparer deux noeuds embarqués contre près de 100 mètres pour de l'ethernet.

Si cette approche peut permettre d'atteindre des performances plus près du gigabit, elle pourrait s'avérer intéressante. Un autre avantage à cette dernière est qu'elle nécessite beaucoup moins de cellules sur la puce reconfigurable, ce qui permettrait

d'instancier davantage de périphériques.

5.2 Adelie/SSI

Pour qu'une nouvelle technologie soit adoptée, il faut en général qu'elle réponde à certaines attentes. En se basant sur le triangle qualité/temps/coût, cette technologie doit donc faire mieux, plus rapidement et à moindre coût. Rares sont les innovations qui réussissent à réunir les trois aspects. Dans ce cas, la somme des avantages doit dépasser la somme des inconvénients.

Dans cette optique, il est important d'évaluer la performance des différents aspects quantifiables d'Adelie/SSI. À cette fin, les paramètres liés au démarrage, à l'initialisation et à la configuration peuvent être évalués en terme de temps et comparés au processus normal.

De par la nature de son fonctionnement, un noeud s'exécutant sous Adelie/SSI troque une part d'espace mémoire contre la facilité de déploiement et de gestion ainsi qu'une robustesse matérielle accrue. Cet espace mémoire peut lui aussi être quantifié.

5.2.1 Démarrage et initialisation

Le temps de démarrage et d'initialisation est une métrique souvent utilisée pour critiquer les performances d'un système d'exploitation. En effet, les ordinateurs sont une des rares composantes électroniques domestiques à nécessiter un temps d'attente si long entre la mise en marche et l'utilisation.

Pour qu'Adelie/SSI soit une technologie viable, il faut donc que les modifications apportées au processus de démarrage et d'initalisation classique ne viennent empirer le cas de l'ordinateur. Différents paramètres peuvent influencer le temps de démarrage sous Adelie/SSI. Pour tenter de mettre en évidence ceux-ci, l'évaluation

du temps de démarrage a été effectuée sur deux systèmes, soit la grappe de calcul hétérogène Thakur ainsi que le réseau de postes de travail Lusk.

De plus, les deux modes de fonctionnement d'Adelie/SSI seront évalués, soit le mode classique par liaison ainsi que le mode expérimental par superposition.

Le temps de démarrage est décomposé en quatre composantes distinctes : le temps d'initialisation du matériel, le temps de chargement et d'exécution du noyau, le temps d'initialisation de l'environnement par Adelie/SSI et finalement le temps de démarrage des autres services.

Le temps d'initialisation du matériel est le temps compris entre le démarrage à froid du système jusqu'avant le chargement du noyau. Ce temps est incompressible logiciellement, si ce n'est que par des ajustements du BIOS.

Le temps de chargement et d'exécution du noyau comprend quant à lui le chargement, sur disque ou via réseau, du noyau et de l'image de système de fichiers initiale s'il y a et leur exécution. La configuration du noyau et donc la taille qui en résulte, les performances du système de disque ainsi que celles du réseau ont un impact à ce niveau.

Le temps d'initialisation de l'environnement par Adelie/SSI est en fait le temps d'exécution du script `adelie-ssi`. Ici, plusieurs paramètres entrent en jeu. La configuration de l'environnement, l'espace occupé par celui-ci sur le serveur et là encore les performances du système de disque ainsi que celles du réseau sont parmi ceux-ci.

Finalement, le temps de démarrage des autres services. Le temps requis par ceux-ci dépend bien évidemment des services à démarrer mais là aussi des performances du système de disque ainsi que celles du réseau.

Section	Serveur (s)	Noeud hôte (s)	Noeud embarqué (s)
Matériel	12.05	12.07	4.07
Noyau	8.33	8.29	8.01
Adelie/SSI	n/a	2.07	3.15
Logiciel	22.33	12.29	5.15
Total	42.71	34.72	20.38

Tableau 5.1 Comparaison des temps de démarrage sur une grappe de calcul hétérogène.

Regardons tout d'abord le tableau 5.1 présentant les performances de la grappe de calcul hétérogène Thakur en mode classique par liaison.

Il est à noté que dans le cas de noeuds embarqués, le noyau se trouve en mémoire flash car les adaptateurs réseau ne supportent pas la norme PXE. De plus, le noyau est un noyau de la branche 2.4, versus un noyau de la branche 2.6 pour le serveur et les noeuds hôtes. Le temps d'initalisation du chargeur PXE ainsi que le téléchargement du noyau sont mininaux dans le cas du noeud hôte.

Le temps d'initialisation de l'environnement Adelie/SSI est extrêmement court dans les deux cas. Cela s'explique par la configuration même de l'image racine principale et de l'image racine destinée aux noeuds embarqués. En effet, ce qui a le plus d'impact sur la performance de l'initialisation de l'environnement est le nombre de fichiers à copier en mémoire locale.

Comme le serveur possède un configuration minimaliste, soit une configuration optimisée pour la performance et ne contient pas de composante graphique, les répertoires à copier sont de taille relativement restreinte. Donc le temps nécessaire à les dupliquer y est proportionnel. Dans le cas de l'environnement des noeuds embarqués, celui-ci n'étant pas partagé avec le serveur, il ne contient que ce qui est nécessaire à ceux-ci.

Le serveur doit offrir bon nombre de services pour permettre à la grappe de fonc-

Section	Tmpfs (s)	Unionfs (s)
Matériel	12.07	12.06
Noyau	8.29	8.00
Adelie/SSI	2.07	13.02
Logiciel	12.29	13.31
Total	34.72	46.38

Tableau 5.2 Comparaison des temps de démarrage d'un noeud de type hôte sur une grappe de calcul hétérogène en utilisant unionfs.

tionner correctement. Compris parmi ceux-ci sont DHCP, Ganglia, NFS, NIS, NTP, PBS, TFTP et SSH. Dans le cas de la grappe de calcul hétérogène, les noeuds ont une configuration minimale, ce qui explique le temps de démarrage des services moindre.

Le tableau 5.2 présente une comparaison entre les temps de démarrage et d'initialisation de la méthode classique par liaison et de la méthode expérimentale par superposition.

Deux points ressortent de cette comparaison. Premièrement, le temps d'initialisation de l'environnement est substantiellement plus long. Bien que le mécanisme par superposition permet d'éviter la copie de nombreux fichiers, des fichiers d'écrasement *(White Out)* doivent être créés pour supprimer les fichiers indésirables.

Dans le cas de la grappe de calcul hétérogène Thakur, le nombre de fichiers à copier dans le mode traditionnel est inférieur au nombre de fichiers à écraser dans le mode expérimental. De plus, il faut tenir compte du chargement du module `unionfs.ko` et de la mise en place des systèmes de fichiers par superposition qui sont relativement substantiels.

On dénote également une légère perte de performance relative au démarrage des services, ce qui s'explique par le fait que l'ensemble des scripts d'initialisation et des

Section	Serveur (s)	Noeud (s)
Matériel	26.07	25.05
Noyau	6.97	16.67
Adelie/SSI	n/a	34.23
Logiciel	34.65	32.09
Total	67.69	108.05

Tableau 5.3 Comparaison des temps de démarrage sur un réseau de postes de travail.

fichiers de configuration ne résident plus en mémoire locale mais sont bien accédés via NFS sur l'image de système de fichiers racine sur le serveur.

Le tableau 5.3 offre une vision différente. Il s'agit du réseau de postes de travail Lusk. Ce sytème partage le segment `drap.polymtl.ca` et possède une configuration plus lourde, offrant entre autre une interface graphique interactive via Gnome ou KDE.

Le chargeur PXE présent sur les noeuds présente un temps de détection beaucoup plus long que celui sur les noeuds hôtes de la grappe de calcul Thakur. Cela explique en partie le temps de chargement du noyau plus long. De plus, le réseau est partagé par plus de noeuds, en plus de tous les autres postes présents sur le segment.

Le nombre d'applications et de services, donc de fichiers de configuration associés, présents pour le fonctionnement du système en mode graphique interactif explique le temps d'initialisation de l'environnement beaucoup plus long avec la méthode classique sur les noeuds. Cela explique également le temps de démarrage des multiples services sur les noeuds.

Le tableau 5.4 offre la contrepartie de la méthode expérimentale par superposition. En effet, le contexte différent de la configuration du réseau de postes de travail Lusk permet de faire ressortir les avantages de cette méthode.

Section	Tmpfs (s)	Unionfs (s)
Matériel	25.05	25.06
Noyau	16.67	16.34
Adelie/SSI	34.23	13.87
Logiciel	32.09	35.77
Total	108.05	91.04

Tableau 5.4 Comparaison des temps de démarrage d'un noeud sur un réseau de postes de travail en utilisant unionfs.

Le temps d'initialisation de l'environnement d'Adelie/SSI se voit pratiquement coupé de plus de la moitié. Cela s'explique par le ratio plus important du nombre de fichiers à copier dans la méthode classique versus le nombre de fichiers à écraser dans la méthode expérimentale.

L'utilisation d'Adelie/SSI permet donc dans certains cas de réduire le temps de démarrage et d'initialisation du système, comme le démontrent les performances de la grappe de calcul Thakur. Même dans les pires conditions, l'augmentation de ce temps n'est pas substantielle, ce temps n'étant que de 40 secondes de plus sur le réseau de poste de travail Lusk.

La méthode expérimentale par superposition démontre un potentiel fort intéressant au niveau du temps d'initialisation. Dans certaines conditions, elle permet même d'obtenir un gain susbtantiel de performance.

5.2.2 Configuration

Un des ajouts principaux d'Adelie/SSI est la facilité avec laquelle une grappe entière peut être déployée et gérée. Par la centralisation de la configuration, il est donc possible de reconfigurer une grappe ou encore d'y déployer des noeuds additionnels en quelques opérations.

Si cette facilité devait être acquise au prix d'un temps élevé, cela viendrait en amoindrir les bénéfices. La performance des mécanismes de configuration est donc une métrique importante pour juger de l'expérience d'administration du système. Les différents modules de configurations seront donc évalués. De plus, l'impact de l'implantation d'un mécanisme de cache est également présenté.

La reconfiguration automatique de la grappe peut se faire au niveau de chaque service, ou encore sur l'ensemble des services. Lorsque plusieurs services sont spécifiés, un mécanisme de résolution des dépendances est employé pour permettre de respecter un certain ordre dans l'appel des modules de configuration.

Le tableau 5.5 présente les temps d'exécution pour des différents modules pour la grappe de calcul hétérogène Thakur. Il faut noter que le temps de résolution des dépendances est calculé pour la résolution de l'ensemble des modules. De plus, ces temps sont obtenus à partir d'une lecture en cache de la configuration.

La configuration présente donc un temps de configuration très acceptable, soit en deçà de 30 secondes. Les versions antérieures du processus de configuration offraient des temps beaucoup plus longs car le fichier de configuration était réinterprété à chaque appel d'un module de configuration.

Un mécanisme de cache a été développé pour réduire ce temps. La signature MD5 ainsi que la taille du ou des fichiers de configuration sont utilisées pour détecter tout

Section	Temps (s)
Calcul des dépendences	7.08
Adelie/SSI	5.60
DHCP	1.99
Ganglia	0.51
NFS	0.88
NIS	2.30
NTP	0.68
OpenSSH	2.60
SysLinux	1.50
Syslog-NG	0.54
Torque	1.36
Total	25.05

Tableau 5.5 Temps d'exécution des différents modules du configurateur sur une grappe de calcul hétérogène.

Section	Temps (s)
Sans cache	5.01
Avec cache	0.04

Tableau 5.6 Impact de la cache sur le temps de chargement de la configuration sur une grappe de calcul hétérogène.

changement à la configuration. C'est seulement lorsqu'un changement est détecté que la configuration doit être réinterprètée.

Le tableau 5.6 indique le temps d'exécution du script principal qui gère la configuration, `adelie-conf-functions.sh`. Le premier appel suivant une modification de la configuration entraîne automatiquement une regénération de la cache.

Comme il est possible de le constater, l'utilisation de la cache permet d'épargner près de cinq secondes par appel. Si l'on considère les dix modules ainsi que dix appels supplémentaires faits par le calcul des dépendances, c'est près de deux minutes qui sont ainsi économisées.

Le réseau de postes de travail Lusk offre des performances sensiblement meilleures.

Section	Temps (s)
Calcul des dépendences	4.45
Adelie/SSI	2.40
DHCP	1.40
Ganglia	0.39
NFS	0.66
NIS	1.68
NTP	0.46
OpenSSH	0.92
SysLinux	1.00
Syslog-NG	0.38
Torque	1.22
Total	14.96

Tableau 5.7 Temps d'exécution des différents modules du configurateur sur un réseau de postes de travail.

Section	Temps (s)
Sans cache	2.60
Avec cache	0.04

Tableau 5.8 Impact de la cache sur le temps de chargement de la configuration sur un réseau de postes de travail.

Le tableau 5.7 présente ces résultats.

Le temps de configuration plus court dans ce contexte s'explique par la performance accrue du processeur ainsi que par celle du système de disques.

Le tableau 5.8 montre les performances au niveau de l'interpétation de la configuration et de la lecture de la cache.

Il est à noter le temps quasi incompressible de la lecture de la cache, qui aide ainsi à amoindrir l'impact des performances locales sur le temps global de configuration.

La reconfiguration d'un système de complexité moyenne nécessite donc un temps d'exécution tout à fait acceptable. Les performances du processeur et des systèmes

Répertoire	Noeud hôte (ko)	Noeud embarqué (ko)
/etc	2244	352
/lib	28	n/a
/media	0	n/a
/tmp	0	0
/var	5516	12
Total	7788	364

Tableau 5.9 Comparaison de l'espace occupé en mémoire vive par les différents répertoires sur une grappe de calcul hétérogène.

de disques ont un impact sur celle-ci. De plus l'utilisation d'un mécanisme de cache permet de réduire de manière substantielle le temps requis à cette configuration.

5.2.3 Espace mémoire

Adelie/SSI possède de nombreux avantages mais aussi quelques inconvénients. L'inconvénient majeur est la perte d'espace mémoire occupé par les répertoires locaux. Cet espace dépend de différents facteurs, principalement le mécanisme employé, que ce soit la méthode traditionnelle par liaison ou encore la méthode expérimentale par superposition.

La méthode par liaison est beaucoup plus coûteuse en terme d'espace que la méthode par superposition. Dans la méthode traditionnelle, les fichiers sont copiés du système de fichier en lecture seule vers un espace en mémoire locale accessible en écriture. Bien que certains fichiers sont exclus de cette copie, l'espace occupé résultant est non-négligeable.

Le tableau 5.9 présente les résultats par répertoire pour la grappe de calcul hétérogène Thakur selon la méthode classique par liaison.

Comme il est possible de le constater, l'espace mémoire occupé par les fichiers

Répertoire	Tmpfs (ko)	Unionfs (ko)
/etc	2244	16
/lib	28	0
/media	0	0
/tmp	0	0
/var	5516	132
Total	7788	148

Tableau 5.10 Comparaison de l'espace occupé en mémoire vive par les différents répertoires sur une grappe de calcul hétérogène en utilisant unionfs.

locaux après le démarrage sur un noeud hôte est d'environ 8 Mo, soit un peu moins de deux pourcents de la capacité totale en mémoire. Pour ce qui est du noeud embarqué, cet espace descend à 364 Ko, soit un peu plus d'un pourcent de la capacité totale de mémoire.

Le tableau 5.10 montre les résultats comparatifs entre la méthode classique par liaison et la méthode expérimentale par superposition sur un noeud hôte. La méthode par superposition n'est pas implémentée sur les noeuds embarqués car un des modules nécessaire à son fonctionnement nécessite plus de 1500 Ko, ce qui vient annuler tout gain possible par l'utilisation de cette technique.

Il est possible de remarquer que sous cette configuration, l'espace mémoire occupé initialement descend sous le seuil des 150 Ko, ce qui est fort appréciable. Il est important de noter que ceci reflète l'espace occupé après de démarrage. La taille de celui-ci peut croître en fonction de l'usage du système et ce indépendamment du mécanisme utilisé.

Le réseau de postes de travail Lusk présente des résultats encore plus éloquents. En regardant le tableau 5.11, il est possible de remarquer la quantité de mémoire impressionnante requise par la méthode classique par liaison : un peu moins de 50 Mo. Cela représente près de cinq pourcents de la capacité totale de chaque noeud.

Répertoire	Tmpfs (ko)	Unionfs (ko)
/etc	37976	16
/lib	28	0
/media	0	0
/tmp	0	0
/var	12240	220
Total	50244	236

Tableau 5.11 Comparaison de l'espace occupé en mémoire vive par les différents répertoires sur un réseau de postes de travail en utilisant unionfs.

Cela s'explique bien entendu par le nombre élevé de fichiers qui sont copiés localement par cette méthode. C'est ici que l'on peut voir briller la méthode par superposition. Seulement 236 Ko d'espace mémoire est utilisé pour contenir les modifications par rapport au système de fichiers de la couche inférieure.

La perte d'espace mémoire liée à la création d'un espace local sur les noeuds est un effet secondaire inévitable d'utiliser Adelie/SSI. Il a par contre été démontré que des deux méthodes, soit la méthode classique par liaison et la méthode expérimentale par superposition, la seconde cause le moins d'impact lié à la diminution de mémoire disponible au système.

CONCLUSION

Ces travaux ont permis de développer une solution intégrée de calcul de haute performance. Cette solution est la seule solution pour grappe de calcul disponible pour Gentoo Linux. Il s'agit d'un mariage parfait alliant le potentiel d'optimisation de Gentoo Linux à la puissance agrégée d'une grappe de calcul de type Beowulf.

De plus, cette solution a été étendue afin de supporter les grappes composées de matériel hétérogène. Ces modifications ont également permis de supporter un nombre illimité de profils d'utilisation des noeuds compris dans ces grappes. Il est ainsi possible d'avoir des noeuds de soumission, de visualisation, de compilation, de calcul.

Le fruit de plusieurs années d'expérience en matière d'élaboration de grappes de calcul a été transcrit sous forme d'heuristiques. Ces heuristiques ont servi à l'élaboration de modules de configuration. Il est ainsi possible de configurer et de tenir à jour une grappe de calcul entière avec un minimum de connaissance du sujet.

Les performances des modules de configuration ont été évaluées. Bien que sans base commune pour comparaison, les temps d'exécution des différents modules sont d'un ordre de grandeur tout à fait acceptable. De plus, il est possible de réduire le temps total de configuration en n'invoquant que les modules requis.

Un mécanisme de cache a été ajouté afin d'éviter la relecture des fichiers de configuration à chaque appel d'un module de configuration ou d'un outil de gestion. Si aucune modification n'a été apportée à ces fichiers, le temps de lecture à partir de la cache passe de quelques secondes à quelques milisecondes.

Les avantanges de la méthode expérimentale par superposition ont été évalués. Ce mécanisme utilise le système de fichiers unionfs pour superposer des répertoires. Ainsi, des répertoires créés en mémoire vive accessibles en écriture sont superposés aux répertoires exportés via NFS qui ne sont accessibles qu'en lecture.

Cette méthode permet de réduire de beaucoup l'espace utilisé en mémoire vive pour contenir les répertoires. Seules les modifications à la couche inférieure, soit le système racine en lecture seule, sont sauvegardées localement. De plus, comme aucun fichier n'est copié lors de la mise en place de l'environnement, le temps d'initialisation du système peut s'en voir réduit.

Lorsque le nombre de fichiers à conserver dans les répertoires locaux est faible, ce mécanisme peut s'avérer plus lent à l'initialisation. En effet, le chargement du module et la mise en place du pseudo système de fichiers nécessite un temps non-négligeable. Par contre, lorsque ce nombre est élevé, les économies de temps sont proportionnelles.

Les limitations en ce qui a trait aux différents mécanismes de communication au sein de la grille composée des plateformes embarquées ont également été mises en évidence.

En effet, il a été démontré que les performances des interfaces gigabit ethernet étaient inférieures aux performances attendues pour des interfaces fast ethernet. De plus, celles-ci présentaient une asymétrie au point de vue de la bande passante entre l'envoi et la réception. Ceci a été expliqué par les limitations au point de vue de la cadence du système ainsi que par le choix de la pile TCP/IP utilisée.

Il a aussi été démontré que le mécanisme de communication par mémoire partagée utilisé pour communiquer entre deux plateformes se trouvant sur la même puce n'était pas une solution viable. En effet, celui-ci n'est pas optimal pour des messages

de courte taille, la bande passante maximum n'étant atteinte que pour de très longs messages. De plus, cette bande passante est bien en deçà des performances pour les adaptateurs ethernet.

Il existe de nombreuses avenues de développement possibles à partir de ce projet. En voici quelques unes qui s'appliquent plus particulièrement au matériel et d'autres au logiciel.

Les interfaces de communication de la plateforme de développement d'Amirix offrent des performances décevantes. L'utilisation d'un protocole plus léger que TCP/IP, couplé à une interface plus simple et plus performante que le gigabit ethernet serait intéressant. Il existe des implémentations de référence pour des protocoles légers à utiliser pour lier deux serdes avec un minimum de matériel. De plus, les nouvelles versions des puces Virtex possèdent des SER/DES beaucoup plus rapides.

Avec plus de cellules logiques disponibles, il serait possible d'implémenter des modules d'optimisation matérielle. Puisque ces puces supportent la reconfiguration partielle, il serait intéressant de développer une librairie de modules qui pourraient être intégrés à la plateforme afin d'accélérer les calculs. Il serait ainsi possible d'avoir des modules prédéfinis qui seraient couplés avec des librairies logicielles classiques tel que BLAS, FFT et plusieurs autres. Il en serait ainsi totalement transparent de la part de l'utilisateur.

La solution de gestion et de déploiement de grappes de calcul est fortement couplé à Gentoo Linux. En effet, des modifications ont été intégrées à Gentoo Linux afin de permettre un fonctionnement transparent. Bien que la plupart des outils et des modules de configuration pourraient être adaptés pour d'autres distributions, ceci nécessiterait des modifications substantielles.

L'ensemble des modules de configuration et des outils sont écrits en script Bash. Cela ne pose pas de problèmes de performance pour une grappe de calcul de taille moyenne mais pourrait poser des problèmes sur une grappe comprenant plusieurs centaines de processeurs.

Les données sont conservées sous forme de tableaux de variables sous Bash. Utiliser une base de donnée permettrait de s'assurer de la cohérence de la configuration lors de modifications locales.

La majorité des modules de configuration n'interprètent pas les données présentes dans les fichiers de configuration qu'ils modifient. En ajoutant plus d'intellingence à ces modules, il serait possible d'intégrer les paramètres de configuration déjà existant à la configuration globale. De plus, ceci permettrait d'éviter d'avoir des configurations contenant des conflits.

Un des problèmes liés à la mise à l'échelle d'Adelie/SSI est son point de défaillance unique, c'est-à-dire le serveur. En effet, si le serveur tombe en panne, toute la grappe tombe en panne également. Il pourrait être intéressant d'explorer une configuration plus robuste. Une solution est proposée : utiliser un bassin de serveurs qui se partageraient les noeuds. La figure VI.1 de l'annexe VI présente un tel système.

Tous les serveurs seraient reliés entre eux par un SAN (Storage Area Network). L'image du système serait partagée à l'aide d'un système de fichiers à stockage distribué, tel que GFS (Global File System). Ce type de système de fichiers supporte les accès concurrents sur un même medium et ce au niveau du disque. Seul le serveur principal aurait accès en écriture à cette partition distribuée. Tous les autres serveurs emploieraient un mécanisme similaire à celui des noeuds. Si un serveur secondaire tombe en panne, un autre serveur peut alors gérer ses noeuds.

Si le serveur principal tombe en panne, un serveur secondaire assumerait le rôle de serveur principal.

Les supercalculateurs amorcent une nouvelle étape de leur évolution. Les premiers systèmes de calcul de haute performance intégrant des puces de type FPGA dans leur architecture sont apparus sur le marché. Pour l'instant, ces systèmes sont fortement couplés aux machines qui les hébergent. Avec l'avancement de la performance couplé à une intégration de plus en plus grande, le même problème qui s'est posé jadis risque de refaire surface. Il faut cesser de voir ces composantes comme des ressources mais bien comme des entités distinctes si l'on désire éviter les problèmes de contention.

RÉFÉRENCES

AMIRIX SYSTEMS (2004a). *Gigabit Ethernet 2D-Mesh System Users Guide.* AMIRIX Systems, Halifax, Nova Scotia.

AMIRIX SYSTEMS (2004b). *PCI Platform FPGA Development Board Users Guide.* AMIRIX Systems, Halifax, Nova Scotia.

AMIRIX SYSTEMS (2004c). *Quad Gigabit Ethernet Platform Detailed Design Document.* AMIRIX Systems, Halifax, Nova Scotia.

ANDERSEN, E. BusyBox. [En ligne]. `http://www.busybox.net/`.

BAR, M. and KNOX, B. (2004). Openmosix: The other kind of hpc cluster. *ClusterWorld.*

BARAK, A. and LA'ADAN, O. (1998). The MOSIX multicomputer operating system for high performance cluster computing. *Journal of Future Generation Computer Systems.*

BOOKMAN, C. (2002). *Linux Clustering: Building and Maintaining Linux Clusters.* Sams Publishing.

CALLAGHAN, B., PAWLOWSKI, B., and STAUBACH, P. (1995). *RFC 1813 – NFS Version 3 Protocol Specification.* Network Working Group.

CRAY (2004). *Cray XD1 Datasheet.* Cray.

CROFT, B. and GILMORE, J. (1985). *RFC 951 – Bootstrap Protocol.* Network Working Group.

D. H. BROWN ASSOCIATES (2004). *Cray XD1 Brings High-Bandwidth Supercomputing to the Mid-Market.* Cray.

DAHMANI, M., MORIN, B., and ROY, R. (2004). Performance evaluation for neutron transport application using message passing. In *Proceedings of the 18th Annual International Symposium on High Performance Computing Systems and Applications*, Winnipeg, Manitoba.

DE LIGNERIS, B. and GIRALDEAU, F. (2003). Thin-OSCAR: Design and future implementation. In *Proceedings of the First OSCAR Symposium*, Sherbrooke, Québec.

DE LIGNERIS, B., SCOTT, S., NAUGHTON, T., and GORSUCH, N. (2003). Open Source Cluster Application Resources (OSCAR): Design, Implementation and Interest for the [Computer] Scientific Community. In *Proceedings of the First OSCAR Symposium*, Sherbrooke, Québec.

DRC COMPUTER CORPORATION (2006). *DRC Coprocessor Modules Datasheet*. DRC Computer Corporation.

DROMS, R. (1993). *RFC 1534 – Interoperation Between DHCP and BOOTP*. Network Working Group.

FINLAYSON, R. (1984). *RFC 906 – Bootstrap Loading using TFTP*. Network Working Group.

GARBER, L. (2005). Researchers build reconfigurable supercomputer. *IEEE Computer*.

GENTOO FONDATION. Gentoo Linux – Gentoo Linux News. [En ligne]. `http://www.gentoo.org/`.

GENTOO-WIKI. Embedded Gentoo - Gentoo Linux Wiki. [En ligne]. `http://gentoo-wiki.com/Embedded_Gentoo`.

GENTOO-WIKI. TinyGentoo - Gentoo Linux Wiki. [En ligne]. `http://gentoo-wiki.com/TinyGentoo`.

GROPP, W., LUSK, E., and STERLING, T. (2003). *Beowulf Cluster Computing with Linux*. The MIT Press, Cambridge, Massachusetts, second edition.

INTEL CORPORATION (1999). *Preboot Execution Environment(PXE) Specification*. Intel Corporation.

JONES, R. (1996). *Netperf: A Network Performance Benchmark*.

KEGEL, D. Building and Testing GCC/GLIBC Cross Toolchains. [En ligne]. `http://kegel.com/crosstool/`.

KOPLA, H. and DALY, P. W. (1999). *A Guide to LaTeX : Document Preparation for Beginners and Advanced Users*. Addison–Wesley Professional, Reading, Massachusetts, third edition.

LEHRTER, J. M., ABU-KHZAM, F. N., BOULDIN, D. W., LANGSTON, M. A., and PETERSON, G. D. (2002). On special-purpose hardware clusters for high-performance computational grids. In *International Conference on Parallel and Distributed Computing and Systems*, Taiwan, Chine.

MALKIN, G. and HARKIN, A. (1998a). *RFC 2348 – TFTP Blocksize Option*. Network Working Group.

MALKIN, G. and HARKIN, A. (1998b). *RFC 2349 – TFTP Timeout Interval and Transfer Size Options,*. Network Working Group.

MEASUREMENT AND CONVERSION SOFTWARE. M.A.C Software - SnapTimePro. [En ligne]. `http://www.measureandconvert.com/prod01.htm`.

PAPADOPOULOS, P. M., KATZ, M. J., and BRUNO, G. (2001). Npaci rocks: Tools and techniques for easily deploying manageable linux clusters. In *Proceedings of the 3rd IEEE International Conference on Cluster Computing*, Newport Beach, California.

PENGUTRONIX. Pengutronix - PTXdist. [En ligne]. `http://www.pengutronix.de/software/ptxdist_en.html`.

PLUMMER, D. C. (1982). *RFC 826 – An Ethernet Address Resolution Protocol.* Network Working Group.

REYNOLDS, J. (1993). *RFC 1497 – BOOTP Vendor Information Extensions.* Network Working Group.

SARUP, M. A Look at System V Initialization. [En ligne]. `http://www.freeos.com/articles/3243/`.

SCYLD SOFTWARE (2004). *Scyld Beowulf Clustering for High Performance Computing.* Scyld Software.

SLOAN, J. (2004). *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI.* O'Reilly, Sebastopol, California.

SOLLINS, K. (1992). *RFC 1350 – The TFTP Protocol (Revision 2).* Network Working Group.

ST. LAURENT, A. M. (2004). *Undestanding Open Source and Free Software Licensing.* O'Reilly, Sebastopol, California.

STERLING, T., SALMON, J., BECKER, D. J., and SAVARESE, D. F. (1999). *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters.* The MIT Press, Cambridge, Massachusetts.

STERLING, T. L., SAVARESE, D., BECKER, D. J., DORBAND, J. E., RANAWAKE, U. A., and PACKER, C. V. (1995). Beowulf: A parallel workstation for scientific computation. In *Proceedings of the 24th International Conference on Parallel Processing*, Boca Raton, Florida.

SUN MICROSYSTEMS (1989). *RFC 1094 – NFS: Network File System Protocol Specification.* Network Working Group.

SYNECTIVE LABS (2004). *Pure Computational Power.* Synective Labs.

THE FREEBSD PROJECT. About FreeBSD Ports. [En ligne]. `http://www.freebsd.org/ports/`.

TIMESYS CORPORATION. Welcome to TimeSys. [En ligne]. `http://www.timesys.com/`.

VERIDAN SYSTEMS (2001). *Portable Batch System Administrator Guide.* Veridian Systems, Mountain View, California.

VRENIOS, A. (2002). *Linux Cluster Architecture.* Sams Publishing.

WALKER, B. J. (2003). *Open Single System Image (openSSI) Linux Cluster Project.* Hewlett–Packard.

WAREWULF CLUSTERING SOLUTION (2005). Warewulf Project. [En ligne]. `http://www.warewulf-cluster.org/cgi-bin/trac.cgi`.

WILLIAMS, T. and THOMAS, C. (2004). *Gnuplot: An Interactive Plotting Program.*

XILINX (2002). *Virtex-II Pro Platform FPGA Handbook.* Xilinx.

ANNEXE I

SCRIPT D'ÉVALUTION DE PERFORMANCE TCP/IP

Le script utilisé pour obtenir les résultats de l'évaluation de la bande passante TCP/IP à l'aide de *NetPerf* est reproduit ici.

```
#!/bin/sh

netperf_root="/opt/netperf-2.3"

host="${1}"
if [ -z "${2}" ]; then
        first=1
else
        first=${2}
fi
if [ -z "${3}" ]; then
        last=8388608
else
        last=${3}
fi

retval=255
while [ ${retval} -eq 255 ]; do
        ssh ${host} start-stop-daemon --start --background --exec ${ ↩
            netperf_root}/netserver
        retval=${?}
done

sleep 1

i=${first}
while [ ${i} -le ${last} ]; do
        if [ ${i} -eq ${first} ]; then
        ${netperf_root}/netperf -l -$(expr ${i} \* 50) -H ${host} -t TCP_STREAM ↩
             -- -m ${i} -s 32768 -S 32768
        else
        ${netperf_root}/netperf -l -$(expr ${i} \* 50) -H ${host} -t TCP_STREAM ↩
```

```
                -- -m ${i} -s 32768 -S 32768 | tail -n 1
        fi

        i=$(expr ${i} \* 2)
done

retval=255
while [ ${retval} -eq 255 ]; do
        ssh ${host} start-stop-daemon --stop --exec ${netperf_root}/netserver
        retval=${?}
done
```

Listage I.1 Script d'évaluation de la bande passante au niveau de la couche TCP/IP.

ANNEXE II

ÉVALUATION DE PERFORMANCE TCP/IP

Les mesures de la bande passante TCP/IP entre les noeuds de la grappe de calcul Thakur ont été ontenues à l'aide du logiciel d'étalonnage *NetPerf*. Les résultats ont été obtenus pour des messages de tailles comprises entre 1 o et 8 Mo. Pour chaque taille de message, le transfert est répété 50 fois et le temps moyen utilisé pour calculer la bande passante.

Taille (octet)	Commutateur (Mo/s)
1	2.30
2	4.57
4	8.16
8	14.55
16	20.98
32	29.91
64	47.41
128	63.84
256	76.59
512	84.28
1024	88.87
2048	90.51
4096	92.58
8192	93.35
16384	93.69
32768	93.83
65536	93.92
131072	94.03
262144	94.05
524288	94.08
1048576	94.09
2097152	94.10
4194304	94.10
8388608	94.10

Tableau II.1 Détails de la performance TCP entre noeuds hôtes sur une grappe de calcul hétérogène.

Taille (octet)	Commutateur (Mo/s)	Direct (Mo/s)	Mémoire partagée (Mo/s)
1	0.16	0.18	0.07
2	0.34	0.39	0.13
4	0.70	0.65	0.24
8	1.50	1.56	0.58
16	2.98	2.98	1.12
32	5.76	5.60	2.44
64	10.28	10.31	5.25
128	13.75	15.11	8.45
256	22.78	20.91	12.81
512	25.78	27.43	16.45
1024	43.66	41.23	21.02
2048	45.35	56.17	24.50
4096	67.34	71.76	27.37
8192	61.72	69.29	29.23
16384	67.41	66.15	30.89
32768	70.66	73.86	31.87
65536	73.68	75.67	32.24
131072	73.84	77.78	32.75
262144	75.40	77.27	32.84
524288	74.64	77.95	32.89
1048576	75.84	77.84	32.92
2097152	75.08	78.78	32.96
4194304	76.05	77.62	32.92
8388608	75.09	78.71	32.94

Tableau II.2 Détails de la performance TCP entre noeuds embarqués sur une grappe de calcul hétérogène.

Taille (octet)	Noeud hôte → embarqué (Mo/s)	Noeud embarqué → hôte (Mo/s)
1	0.34	0.15
2	0.89	0.27
4	1.34	0.55
8	3.29	1.15
16	4.93	2.44
32	9.65	4.74
64	15.54	9.60
128	27.78	15.65
256	32.90	19.59
512	44.42	22.11
1024	56.94	32.09
2048	65.88	34.34
4096	75.32	45.75
8192	83.54	51.34
16384	89.29	55.73
32768	91.25	61.01
65536	92.73	60.59
131072	93.39	63.13
262144	93.72	62.41
524288	93.90	64.61
1048576	93.98	64.61
2097152	94.05	65.26
4194304	94.08	64.99
8388608	94.09	63.40

Tableau II.3 Détails de la performance TCP entre noeuds hôtes et embarqués sur une grappe de calcul hétérogène.

ANNEXE III

TEMPS DE DÉMARRAGE

Les mesures du temps de démarrage ont été obtenues à l'aide du chronographe numérique *SnapTimePro*. Ce chronographe permet de calculer des sous-intervals. La commande time a aussi été utilisée pour obtenir le temps intitulé Adelie/SSI. Les mesures ont été répétées à 4 reprises, de manière consécutives. La première mesure est toujours éliminée.

Le temps intitulé matériel débute au démarage à froid du système et se termine immédiatement avant le chargement du noyau sur le serveur ou le chargement du client PXE sur les noeuds. Le temps intitulé noyau débute lors du chargement du noyau sur le serveur ou du client PXE sur les noeuds. Le temps intitulé Adelie/SSI est le temps d'exécution de la fonction start du script d'initialisation adelie-ssi. Le temps intitulé Logiciel débute à la fin de l'exécution du script d'initialisation adelie-ssi et se termine à l'apparition de l'invite de commande.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Matériel	12.07	12.04	12.03	12.05
Noyau	7.98	8.05	8.97	8.33
Adelie/SSI	n/a	n/a	n/a	n/a
Logiciel	21.99	23.00	22.01	22.33
Total	42.04	43.09	43.01	42.71

Tableau III.1 Temps de démarrage du serveur sur une grappe de calcul hétérogène.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Matériel	12.08	12.05	12.08	12.07
Noyau	7.95	8.00	8.92	8.29
Adelie/SSI	2.07	2.07	2.07	2.07
Logiciel	11.98	12.93	11.97	12.29
Total	34.08	35.05	35.04	34.72

Tableau III.2 Temps de démarrage d'un noeud de type hôte sur une grappe de calcul hétérogène.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Matériel	4.07	4.06	4.08	4.07
Noyau	8.01	8.01	8.01	8.01
Adelie/SSI	3.15	3.15	3.16	3.15
Logiciel	4.84	4.85	5.76	5.15
Total	20.07	20.07	21.01	20.38

Tableau III.3 Temps de démarrage d'un noeud de type embarqué sur une grappe de calcul hétérogène.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Matériel	12.07	12.05	12.05	12.06
Noyau	7.99	8.00	8.02	8.00
Adelie/SSI	13.03	13.00	13.02	13.02
Logiciel	12.98	13.03	13.91	13.31
Total	46.07	46.08	47.00	46.38

Tableau III.4 Temps de démarrage d'un noeud de type hôte sur une grappe de calcul hétérogène en utilisant unionfs.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Matériel	26.07	26.07	26.06	26.07
Noyau	6.97	6.96	6.99	6.97
Adelie/SSI	n/a	n/a	n/a	n/a
Logiciel	33.98	34.98	34.98	34.65
Total	67.02	68.01	68.03	67.69

Tableau III.5 Temps de démarrage du serveur sur un réseau de postes de travail.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Matériel	25.05	25.05	25.05	25.05
Noyau	17.04	15.02	17.96	16.67
Adelie/SSI	34.90	33.95	33.84	34.23
Logiciel	33.04	33.01	30.23	32.09
Total	110.03	107.03	107.08	108.05

Tableau III.6 Temps de démarrage d'un noeud sur un réseau de postes de travail.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Matériel	25.04	25.06	25.07	25.06
Noyau	16.99	17.03	15.01	16.34
Adelie/SSI	13.99	13.72	13.89	13.87
Logiciel	34.00	36.28	37.03	35.77
Total	90.02	92.09	91.00	91.04

Tableau III.7 Temps de démarrage d'un noeud sur un réseau de postes de travail en utilisant unionfs.

ANNEXE IV

CONFIGURATION

Les mesures du temps de configuration pour les différents modules ont été obtenues à l'aide de la commande time. Les mesures ont été répétées à 4 reprises, de manière consécutives. La première mesure est toujours éliminée.

Les temps sont obtenus en utilisant une cache valide. Le temps total comprend l'exécution complète de l'outil *adelie-configure*. Le temps intitulé calcul des dépendances est obtenu par soustraction de la somme des temps des modules du temps total. Le fichier de cache est effacé avant chaque essai lors du calcul du temps de chargement de la configuration sans cache.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Calcul des dépendences	6.99	7.15	7.11	7.08
Adelie/SSI	5.98	5.32	5.50	5.60
DHCP	1.99	1.97	2.00	1.99
Ganglia	0.50	0.51	0.51	0.51
NFS	0.88	0.88	0.89	0.88
NIS	2.22	2.14	2.53	2.30
NTP	0.65	0.75	0.65	0.68
OpenSSH	2.57	2.67	2.57	2.60
SysLinux	1.64	1.44	1.42	1.50
Syslog-NG	0.54	0.54	0.54	0.54
Torque	1.36	1.36	1.37	1.36
Total	25.32	24.73	25.09	25.05

Tableau IV.1 Temps d'exécution des différents modules du configurateur sur une grappe de calcul hétérogène.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Sans cache	5.03	5.01	4.99	5.01
Avec cache	0.04	0.04	0.04	0.04

Tableau IV.2 Temps de chargement de la configuration sur une grappe de calcul hétérogène.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Calcul des dépendences	4.47	4.45	4.43	4.45
Adelie/SSI	2.38	2.43	2.39	2.40
DHCP	1.43	1.38	1.38	1.40
Ganglia	0.39	0.39	0.39	0.39
NFS	0.66	0.66	0.66	0.66
NIS	1.66	1.71	1.66	1.68
NTP	0.46	0.47	0.46	0.46
OpenSSH	0.92	0.92	0.92	0.92
SysLinux	1.00	0.99	1.02	1.00
Syslog-NG	0.38	0.38	0.37	0.38
Torque	1.22	1.22	1.22	1.22
Total	14.97	15.00	14.90	14.96

Tableau IV.3 Temps d'exécution des différents modules du configurateur sur un réseau de postes de travail.

Section	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Moyenne (s)
Sans cache	2.62	2.60	2.59	2.60
Avec cache	0.04	0.04	0.04	0.04

Tableau IV.4 Temps de chargement de la configuration sur un réseau de postes de travail.

ANNEXE V

FICHIER DE CONFIGURATION

Le contenu des fichiers de configuration centrale `/etc/adelie/adelie.conf` pour la grappe de calcul hétérogène Thakur ainsi que le réseau de postes de travail Lusk sont reproduits ici.

```
# /etc/adelie/adelie.conf
#
# Copyright (c) 2004-2005 Cyberlogic. All rights reserved.
# Distributed under the terms of the GNU General Public License Version 2.
#
# History:
#   3.1.0 Benoit Morin <benoit.morin@adelielinux.com>
#   3.0.0 Benoit Morin <benoit.morin@adelielinux.com>
#
# Descprition:
#   Adelie/SSI cluster configuration file.

cluster-name thakur {
        home-server 132.207.162.11;

        nis-domain-name drap.polymtl.ca;
        nis-server 132.207.162.11;

        host-type server {
                processor-type x86;

                default-gateway 132.207.162.1;

                primary-interface eth0;
                dns-domain-name eth0 thakur;
                network-mask eth0 255.255.255.0;

                dns-domain-name eth1 drap.polymtl.ca;
                network-mask eth1 255.255.255.0;

```

```
        host-name thakur {
                mac-address eth0 00:02:B3:46:96:75;
                ip-address eth0 10.0.0.1;

                mac-address eth1 00:10:DC:DB:64:3D;
                ip-address eth1 132.207.162.16;
        }
}

host-type node {
        processor-type x86;
        processor-count 1;

        primary-interface eth0;
        dns-domain-name eth0 thakur;
        network-mask eth0 255.255.255.0;

        host-name node01 {
                mac-address eth0 00:02:B3:87:F7:3B;
                ip-address eth0 10.0.0.2;
        }

        host-name node02 {
                mac-address eth0 00:02:B3:9F:55:B8;
                ip-address eth0 10.0.0.3;
        }

        host-name node03 {
                mac-address eth0 00:02:B3:46:96:7C;
                ip-address eth0 10.0.0.4;
        }

        host-name node04 {
                mac-address eth0 00:02:B3:9F:58:74;
                ip-address eth0 10.0.0.5;
        }
}

host-type ppc {
        processor-type ppc;

        root-path /nfsroot;
```

```
primary-interface gbe0;
dns-domain-name gbe0 thakur;
network-mask gbe0 255.255.255.0;

dns-domain-name gbe1 gbe.thakur;
network-mask gbe1 255.255.255.252;

dns-domain-name seth0 seth.thakur;
network-mask seth0 255.255.255.252;

host-name ppc01 {
        mac-address gbe0 00:50:C2:29:20:60;
        ip-address gbe0 10.0.0.101;
        ip-address gbe1 10.0.2.1; # pointopoint 10.0.2.2
        ip-address seth0 10.0.1.1; # pointopoint 10.0.1.2
}

host-name ppc02 {
        mac-address gbe0 00:50:C2:29:20:62;
        ip-address gbe0 10.0.0.102;
        ip-address gbe1 10.0.4.1; # pointopoint 10.0.4.2
        ip-address seth0 10.0.1.2; # pointopoint 10.0.1.1
}

host-name ppc03 {
        mac-address gbe0 00:50:C2:29:20:44;
        ip-address gbe0 10.0.0.103;
        ip-address gbe1 10.0.2.2; # pointopoint 10.0.2.1
        ip-address seth0 10.0.3.1; # pointopoint 10.0.3.2
}

host-name ppc04 {
        mac-address gbe0 00:50:C2:29:20:46;
        ip-address gbe0 10.0.0.104;
        ip-address gbe1 10.0.4.2; # pointopoint 10.0.4.1
        ip-address seth0 10.0.3.2; # pointopoint 10.0.3.1
}

host-name ppc05 {
        mac-address gbe0 00:50:C2:29:20:5C;
        ip-address gbe0 10.0.0.105;
```

```
                        ip-address gbe1 10.0.6.1; # pointopoint 10.0.6.2
                        ip-address seth0 10.0.5.1; # pointopoint 10.0.5.2
                }

                host-name ppc06 {
                        mac-address gbe0 00:50:C2:29:20:5E;
                        ip-address gbe0 10.0.0.106;
                        ip-address gbe1 10.0.8.1; # pointopoint 10.0.8.2
                        ip-address seth0 10.0.5.2; # pointopoint 10.0.5.1
                }

                host-name ppc07 {
                        mac-address gbe0 00:50:C2:29:20:54;
                        ip-address gbe0 10.0.0.107;
                        ip-address gbe1 10.0.6.2; # pointopoint 10.0.6.1
                        ip-address seth0 10.0.7.1; # pointopoint 10.0.7.2
                }

                host-name ppc08 {
                        mac-address gbe0 00:50:C2:29:20:56;
                        ip-address gbe0 10.0.0.108;
                        ip-address gbe1 10.0.8.2; # pointopoint 10.0.8.1
                        ip-address seth0 10.0.7.2; # pointopoint 10.0.7.1
                }
        }
}
```

Listage V.1 Fichier de configuration central d'une grappe de calcul hétérogène.

```
# /etc/adelie/adelie.conf
#
# Copyright (c) 2004-2005 Cyberlogic. All rights reserved.
# Distributed under the terms of the GNU General Public License Version 2.
#
# History:
#   3.1.0 Benoit Morin <benoit.morin@adelielinux.com>
#   3.0.0 Benoit Morin <benoit.morin@adelielinux.com>
#
# Descprition:
#   Adelie/SSI cluster configuration file.

cluster-name lusk {
        processor-type x86_64;
```

```
15
        default-gateway 132.207.162.1;

        home-server 132.207.162.11;

20      nis-domain-name drap.polymtl.ca;
        nis-server 132.207.162.11;

        primary-interface eth0;
        dns-domain-name eth0 lusk;
25      network-mask eth0 255.255.255.0;

        host-type server {
                processor-count 0;

30              host-name lusk {
                        mac-address eth0 00:0F:B5:42:6E:4E;
                        ip-address eth0 132.207.162.14;
                }
        }
35
        host-type node {
                processor-count 1;

                host-name lab01 {
40                      mac-address eth0 00:11:09:6A:54:44;
                        ip-address eth0 132.207.162.101;
                }

                host-name lab02 {
45                      mac-address eth0 00:11:09:6B:4F:E6;
                        ip-address eth0 132.207.162.102;
                }

                host-name lab03 {
50                      mac-address eth0 00:11:09:6A:54:2B;
                        ip-address eth0 132.207.162.103;
                }

                host-name lab04 {
55                      mac-address eth0 00:11:09:5D:14:EC;
                        ip-address eth0 132.207.162.104;
```

```
                }

                host-name lab05 {
                        mac-address eth0 00:11:09:6A:54:6D;
                        ip-address eth0 132.207.162.105;
                }

                host-name lab06 {
                        mac-address eth0 00:11:09:6A:54:3F;
                        ip-address eth0 132.207.162.106;
                }

                host-name lab07 {
                        mac-address eth0 00:13:D4:D2:89:F9;
                        ip-address eth0 132.207.162.107;
                }

                host-name lab08 {
                        mac-address eth0 00:13:D4:D2:89:A4;
                        ip-address eth0 132.207.162.108;
                }

                host-name lab09 {
                        mac-address eth0 00:13:D4:D2:8B:26;
                        ip-address eth0 132.207.162.109;
                }

                host-name lab10 {
                        mac-address eth0 00:13:D4:D2:8B:1E;
                        ip-address eth0 132.207.162.110;
                }

                include "/etc/adelie/adelie-detect.conf";
        }
}
```

Listage V.2 Fichier de configuration central d'un réseau de postes de travail.

ANNEXE VI

PROPOSITION D'ARCHITECTURE POUR ACCROÎTRE LA ROBUSTESSE D'ADELIE/SSI

Il est proposé d'accroître la robustesse d'Adelie/SSI en utilisant un bassin de serveurs. Voici un diagramme détaillant les couches des systèmes de fichiers utilisés dans cette architecture.

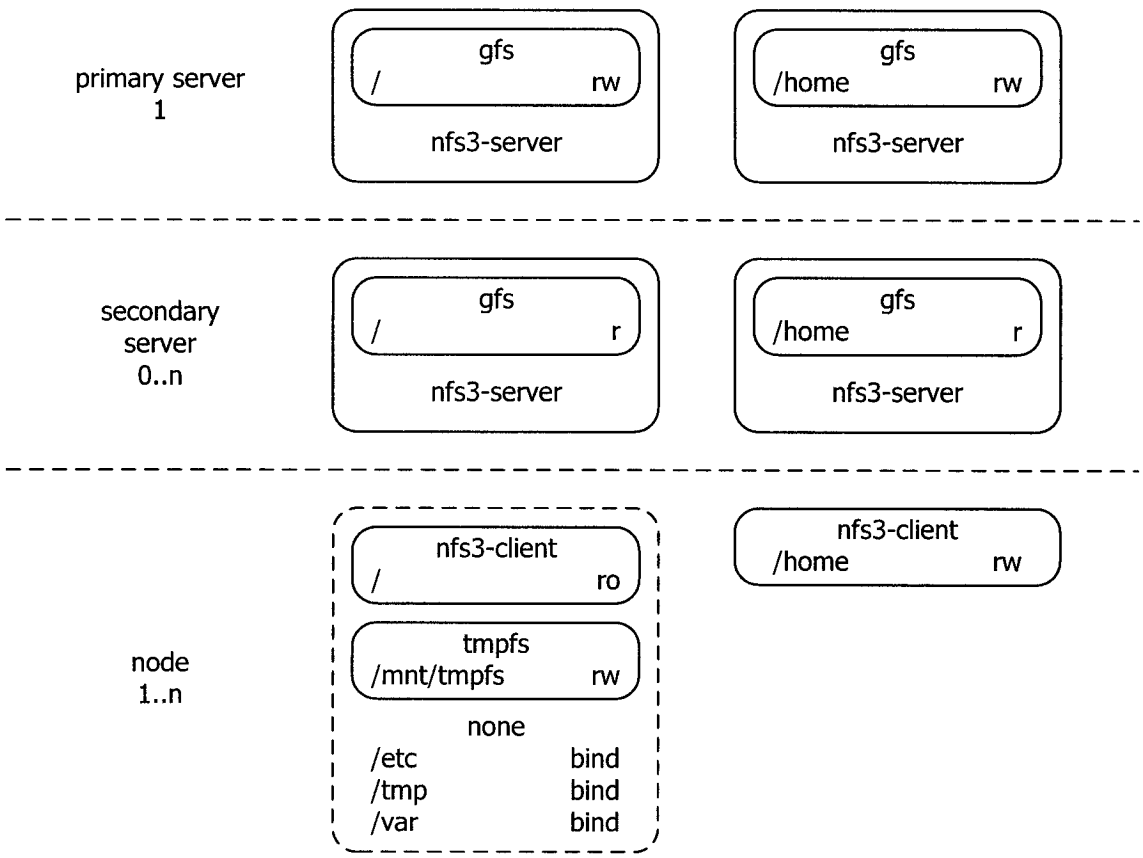

Figure VI.1 Détails des couches de systèmes de fichiers pour un système robuste.

ANNEXE VII

PERFORMANCE EVALUATION FOR NEUTON TRANSPORT APPLICATION USING MESSAGE PASSING

Performance Evaluation for Neutron Transport Application Using Message Passing

M. Dahmani*
Nuclear Engineering Institute
École Polytechnique de Montréal
Montréal, Québec, Canada
E-mail: Mohamed.Dahmani@polymtl.ca
*Corresponding author

B. Morin
Department of Computer Engineering
École Polytechnique de Montréal
Montréal, Québec, Canada
E-mail: Benoit.Morin@polymtl.ca,

R. Roy
Department of Computer Engineering
École Polytechnique de Montréal
Montréal, Québec, Canada
E-mail: Robert.Roy@polymtl.ca

Abstract: In this paper, recent advances in parallel software development for solving neutron transport problems are presented. Following neutron paths along the characteristics of the system, the transport equation is solved to obtain the scalar flux per region and energy group. Due to the excessive number of tracks in the demanding context of 3D large-scale calculations, the parallelization of the solver is considered in order to obtain fast iterative solution. Different load balancing strategies are used for the distribution of the tracks along processors. These strategies are based on the calculation load implied by each track length. The performance of the MPI implementation, using different parallel machines, is analyzed for realistic applications and numerical results are shown. A comparative study of different networks existing on the market and the influence of their parameters on total communication time is also presented.

Keywords: Transport equation; message passing; performance evaluation; networking.

Biographical notes: Mohamed Dahmani is currently a research professional at the Nuclear Engineering Institute at École Polytechnique de Montréal (EPM), Canada. His main interests are reactor physics modeling and benchmarking, along with high performance computing. He received a first Doctorat in 1999, while working at C.E.A. Saclay-France on reactor kinetics methods. Mohamed completed his second Ph.D. in Computer Engineering at EPM in May 2005, the subject being the development of parallel algorithms for large-scale deterministic transport solvers.

1 Introduction

The goal of the neutron transport theory is to determine the distribution of the neutrons in a certain domain (for example: nuclear reactor). In the most general situations, this distribution is a function of three position variables, two angular variables, the neutron energy and the time. With several fundamental physical assumptions, conserva-

tion of neutrons is expressed, in term of their distribution, by the Boltzmann transport equation. The interactions between the neutrons and the atomic nuclei in the domain is expressed by the cross sections defined as a certain probability that a given interaction held. These cross sections are considered as a data for the reactor calculations. The solution of the transport equation in its general form is mathematically impossible. Therefore, several deterministic methods were developed in order to solve the transport equation in particular situations. The most popular methods used are: collision probability, the method of discrete ordinates and the method of characteristics which is the method of our interest in this paper. Different academic and industrial codes based on these methods were developed in many countries all over the world. The recent advances in computer capability in term of memory size and processor speed, on one hand, and the development of the high performance computing methods on the other hand allow investigating the boundaries of the old models in many scientific and engineering applications. Like other applications, the advanced reactor core designs require the advanced computational methods in order to achieve high accuracy in reasonable computational time. In order to solve such a large-scale problems with a minimum amount of CPU time, the parallelization of our solver, based on characteristics method (MCI), is now considered. This parallelization is done by distributing the group of tracking lines on several processors. In this paper, we are interested to evaluate the performance of the parallel program, used in our solver, to solve the 3D transport problems. The message passing implementation using MPI is also presented. Several numerical tests are done to analyze the performance of the algorithm using MPI implementation.

CANDU reactor-physics simulations generally consist of three stages (Roy et al (2004)):

- Calculation of lattice properties for the *bare* CANDU cell, which includes fuel bundle, coolant, pressure tube, gas gap and calandria tube, and the appropriate amount of moderator, but which excludes any interstitial reactivity device;
- Calculation of *incremental* cross sections of reactivity devices, which represent the effect of such devices, and which are added to the basic lattice cross sections in a modelled volume around the reactivity device;
- Modelling of the entire reactor core in three dimensions, and calculation of the core-wide flux and power distributions.

This paper focuses on simulations for the second stage where extensive 3D simulations are done in order to correctly represent the reactivity devices. The movement of these devices influence the neutron population and accurate nuclear properties are essential for safety analysis and core follow-up. Comparisons with site measurements have shown that numerical simulations of the operating CANDUs gives accurate results. However, new Canadian reactor core designs are now evolving from the original CANDU design, and these new design features are more demanding on HPC resources.

An outline of the paper now follows. We begin, in section 2, by presenting a brief review of the neutron transport theory and the method of characteristics used to solve the transport equation. Section 3 is devoted to the presentation of the parallel characteristics solver. In section 4, message passing implementation using the MPI standard and an optimized Linux kernel is described. Then, tests and numerical results pertinent to the field of Canadian nuclear engineering will be shown. Finally, we draw some conclusions.

2 Theoretical background

2.1 Transport equation

The time-independent transport equation can be written as follows (Roy (2003)):

$$\hat{\Omega}\cdot\vec{\nabla}\Phi(\vec{r},\hat{\Omega},E)+\Sigma_t(\vec{r},E)\Phi(\vec{r},\hat{\Omega},E) = \int_0^\infty dE' \int_{4\pi} d^2\Omega'\,\Sigma_s(\vec{r},\hat{\Omega}\leftarrow\hat{\Omega}',E\leftarrow E')\Phi(\vec{r},\hat{\Omega}',E') + \chi(\vec{r},E)\int_0^\infty dE'\,\frac{\nu}{K_{\text{eff}}}\Sigma_f(\vec{r},E')\int_{4\pi} d^2\Omega'\,\Phi(\vec{r},\hat{\Omega}',E'); \quad (1)$$

where

- $\vec{r}$ is a spatial point in the domain D;
- $\hat{\Omega}$ is the solid angle;
- E is the neutron energy;
- $\Phi(\vec{r},\hat{\Omega},E)$ is the angular flux;
- Σ_t is the total cross section;
- Σ_s is the scattering cross section;
- Σ_f is the fission cross section;
- ν is the secondary number of neutrons generated by the fission reactions;
- K_{eff} is the effective multiplication factor;
- χ is the energy-dependent neutron spectra.

After the energy domain discretization using a multi-group formalism, the multi-group isotropic transport equation to be solved to obtain the flux from an isotropic source is

$$\left(\hat{\Omega}\cdot\vec{\nabla}+\Sigma^g(\vec{r})\right)\Phi^g(\vec{r},\hat{\Omega}) = Q^g(\vec{r}); \quad (2)$$

where

- g is the energy group;
- $\Sigma^g(\vec{r})$ is the total cross section in group g;
- $Q^g(\vec{r})$ is the source in group g.

The source in Eq.(2) is composed of two terms, the isotropic fission and scattering sources:

$$Q^g(\vec{r}) = \frac{\chi^g}{4\pi K_{\text{eff}}} \sum_{g'} \nu \Sigma_f^{g'}(\vec{r}) \phi^{g'}(\vec{r}) + \frac{1}{4\pi} \sum_{g'} \Sigma_s^{g \leftarrow g'}(\vec{r}) \phi^{g'}(\vec{r}); \quad (3)$$

where $\phi^g(\vec{r})$ is the scalar flux in group g.

2.2 Solution of the transport equation with characteristics method

The main idea behind the characteristics method is to solve the differential form of the transport equation following the straight lines (characteristics or tracking lines) (Askew (1972)). The neutron trajectories will be followed in the local coordinates system where an observer is traveling in the neutron direction. The basic transport operator is then transformed into a total differential operator. The local multi-group characteristics equation is then given by

$$\left(\frac{d}{ds} + \Sigma^g(\vec{r} + s\hat{\Omega})\right) \Phi^g(\vec{r} + s\hat{\Omega}, \hat{\Omega}) = Q^g(\vec{r} + s\hat{\Omega}); \quad (4)$$

where s stands for the variable along the characteristics line. The spatial and the angular domains are sampled using a ray tracing procedure as described in Dahmani(a) et al (2002). The Υ domain is first covered by choosing a quadrature set of solid angles, composed of discretized directions and their corresponding weights $(\hat{\Omega}_i, \omega_i)$. Then, the plane $\Pi_{\hat{\Omega}_i}$ perpendicular to any selected direction i, is split into a Cartesian grid meshing and the starting points $\vec{p}_{i,n}$ of each characteristics are found. For each discretized direction i, a whole set of tracks $\vec{T}_{i,n}$ will be generated. When traveling across different regions, the neutron beam following the characteristics crosses $K_{i,n}$ segments numbered by index k; the segment lengths are labeled L_k and the region numbers are N_k. Assume that the segment k crosses region j, then the local relationship between the incoming and outgoing angular flux is given by

$${}^{\text{out}}\phi_j^g(k) = {}^{\text{in}}\phi_j^g(k) + \left[\frac{Q_j^g}{\Sigma_j^g} - {}^{\text{in}}\phi_j^g(k)\right] E(\Sigma_j^g L_k); \quad (5)$$

where $E(x) = 1 - e^{-x}$.

The average scalar flux in region j is calculated using

$$\Phi_j^g = \frac{Q_j^g}{\Sigma_j^g} + \frac{1}{\Sigma_j^g V_j} \sum_i \omega_i \sum_n \pi_{i,n} \sum_k \delta_{jN_k} \Delta_{i,n,k}^g; \quad (6)$$

where $\Delta_{i,n,k}^g = {}^{\text{in}}\phi_j^g(k) - {}^{\text{out}}\phi_j^g(k)$ is the flux difference on segment k and $\pi_{i,n}$ is the weight associated with track $\vec{T}_{i,n}$.

From this equation, we see that the iteration sweep over all characteristics involves the sequential evaluation of outgoing fluxes of Eq. (5) on every single track and the adding of flux differences $\Delta_{i,n,k}^g$ for each track $\vec{T}_{i,n}$ in Eq.(6).

3 Parallel characteristics solver

3.1 Finest granularity level

In a parallel characteristics solver, the basic computation unit is focused on repeating the same sequence of calculations on each tracking line. The characteristics sweep on one single line represents the atomic level for the solver from which we can construct the overall solution for each region, each angle, and each energy group. As we have seen before, the group variable can easily be treated using data parallelism. We now define the finest granularity level of operations that can be performed without any communication from the point of view of task parallelism. At this stage, the required local data is a collection of segment lengths L_k and the region numbers encountered along the line. The nuclear data needed are the local total cross section Σ_j^g and the scattering cross section from one group to itself $(\Sigma_s^{g \leftarrow g})_j$. No scattering matrices are necessary; this approach is important to preserve certain linearity in calculation and to ensure independence between the calculations performed on different tracks. The global solution is then the combination of these atomic level solutions.

In real world applications, the number of tracks is much greater than the number of processors available. The parallelization of the solver is then done by grouping the tracks and each slave processor takes control of one of these groups. At each iteration step, a reduce/broadcast operation is activated to recover the partial contribution to the angular fluxes; each processor has its own copy of the flux and the source arrays with a consistent unknown ordering. The scalar flux is then reconstructed on every single processor before the iteration procedure starts (multicasting communication process) .

3.2 Distribution of the tracks and load balancing

We now study the mechanism by which the atomic tasks are assigned to run on the different physical processors. The term *process* refers to a processing or computing agent that performs tasks (Grama et al (2003)); this abstract entity is more convenient to study task-dependency and task-interaction without having to adhere to a rigorously defined parallel architecture. Processes can then run on physical processors. The allocation of tasks to processes is

done by a *mapping*:

$$\begin{aligned} M : \mathcal{T} &\rightarrow \mathcal{P} \\ (i,n) &\mapsto p \end{aligned}$$

from $\mathcal{T}$ the domain of characteristics onto $\mathcal{P}$ the range of processes. As explained above, the tracking lines $\vec{T}_{i,n}$ are identified by their direction i and their starting point n. A global numbering of tracks has no particular use, so we will assume here that tracks are numbered from 1 to N_T in the order they are generated. This global numbering scheme $l(i,n)$ will be assumed when needed in the following.

Since the calculation time for a track linearly depends on its number of segments, the process calculation load defined as

$$L(p) = \sum_{(i,n)\in M^{-1}(p)} t_T(i,n); \tag{7}$$

is not necessarily uniform even if we put the same number of tracks on each process. An evenly number of segments distributed on each processor will be the major criteria to build our load balancing algorithm. A good mapping will take into account the total calculation load of each batch of tracks. The options currently implemented to distribute tracks in the MCI module are:

SPLT Distribution of beams of tracks: in this option, we simply subdivide the tracks in batches of $\frac{N_T}{N_P}$ tracks, as these are generated by the EXCELT module. The first process will work on the first $\frac{N_T}{N_P}$ generated, the second process takes the next $\frac{N_T}{N_P}$ and so on. For domains that contain a lot of small regions, one of these track batches may contain larger number of segments than other ones.

STRD Round-robin distribution: the tracks are distributed in the sequential order of processes coming back to the first when all processors have been given a batch. It is a cyclic mapping of all tracks to a single processor, such that $p \equiv l(i,n) \bmod N_P$. The first process takes the tracks numbered $1, 1+N_P, 1+2N_P, \ldots$, the second process $2, 2+N_P, 2+2N_P, \ldots$, etc. Using this option, it is statistically unexpected to have two batches of tracks having a substantial difference in the average number of segments;

ANGL Distribution on angles: all the tracks having the same direction are grouped together. So, each process takes the tracks with the same direction, so the mapping is $p \equiv i$ or $M(i,n) = i$. Because the number of tracks generated in one direction is not necessarily the same as in other directions, this option may lead to imbalance the load. This option also limits the number of processes to be a multiple of the number of angles.

Several tests were done to show that the STRD option is more balanced than the two other options. Each processor is loaded almost evenly, and the average number of segments on each processor is almost the same (Dahmani(b) et al (2003)).

3.3 Parallel solution of the transport equation

Once a static load balancing scheme is chosen, there exists a mapping $p = M(i,n)$ that gives for any characteristics $\vec{T}_{i,n}$ the process p where computation is performed. Two computational approaches for distributing the groups of tracks over the processors can be considered. The first approach is that one processor (*master*) distributes the tracks to other processors (*slaves*); the second approach is that every processor generates tracks by calling the EXCELT module and takes its own batch of tracks. In our implementation, we do not use the first approach because when the master processor is working to generate the tracks, the slaves remain in wait status. In addition to that, because the size of the tracking file is generally quite large, the communication times become non negligible. With the second approach, we can avoid to have the processors in wait and also to reduce the communications.

At the solver level, each processor takes control of batches of tracks to accumulate contributions to average angular fluxes; a partial sum of flux differences is accumulated by region and energy group.

$$\Delta_p \Phi_j^g = \frac{1}{\Sigma_j^g V_j} \sum_{(i,n)\in M^{-1}(p)} \omega_i \pi_{i,n} \sum_k \delta_{jN_k} \Delta_{i,n,k}^g. \tag{8}$$

At each inner iteration, the processors communicate their results to all other processes by using a reduce/broadcast procedure. This total reduction operation is necessary to add all the contributions together in order to construct the scalar flux by region and energy group from the partial sums of Eq. (8).

$$\Phi_j^g = \frac{Q_j^g}{\Sigma_j^g} + \sum_P \Delta_p \Phi_j^g. \tag{9}$$

Each process, working on its own group of tracks, executes the following iterative scheme:

1. Guess new fission sources and incoming current (outer loop);
   (a) Guess scattering sources (inner loop);
   (b) Compute the flux for each energy group:
      i. compute local solution for each characteristics line;
      ii. perform a reductive sum of all contributions to flux;
      iii. apply global inner acceleration technique.
   (c) If flux map are not converged, go to (a).
2. Apply global outer acceleration techniques;
3. Compute the K_{eff} for next neutron generation;
4. If not converged, go to 1.

After the reduction sum operation, at the end of the inner loop, all processes have the same copy of the flux.

3.4 Related work

The parallelization of the transport equation, based on Collision Probability method, obtained after distributing the energy variable in multi-group solver, as well as the usual spatial decomposition methods, have already been developed a few years ago (Quaddouri et al (1996)). Domain decomposition techniques have also been explored for large-scale transport calculations using the method of characteristics. The spatial decomposition of multi-assembly problems where neutron paths are directly connected when crossing assemblies exhibit limited speedup (5.3 on 8 processors) on shared memory Sun Enterprise4000 because of the tight coupling between the assemblies (Kosaka and Saji (1999)). The angular decomposition technique, where directions are distributed among a set of processors, has also been tested (Lee et al (2000)). Speedup obtained using this angular decomposition is of the order of 3.7 for 4 processors, and 6.8 for 8 processors. Other works were done in this context (Azmy (1993); Sjoden and Haghighat (1997)). In these works, based on S_N method, the domain decomposition techniques are used for the spatial and/or the angular variables.

All these techniques use the domain decomposition methods. No explicitely load balancing techniques were applied.

4 Implementation of the message passing

In this section, we give details about the operating system and the implementation of the parallel characteristics solver.

4.1 MPI Implementation

To perform the communications between processes participating in calculation, we use a Message Passing Interface (MPI) (Gropp et al (1994)). All MPI communications require a communicator and the MPI processes can only communicate if they share the communicator. The standard MPI includes a whole set of routines that can be used for collective communications. When implementing the MCI solver, we used the variant of the reduce operations where the result is returned to all processes in the communicator. On that case, the standard specification requires that all processes participating in these operations receive identical results using a single Fortran call:

```
call MPI_ALLREDUCE(totF,partF,Nr*Ng,
$ MPI_DOUBLE_PRECISION, MPI_SUM,
$ MPI_COMM_WORLD,ierr)
```

equivalent to the following C instruction:

```
MPI_Allreduce(partF,totF,Ng*Nr,MPI_DOUBLE,
      MPI_SUM,MPI_COMM_WORLD);
```

Note that these all-reduce operations can be transformed into a one-node reduce, followed by a broadcast of the resulting values. However, a better performance is obtained if we take into account the possibility to send several synchronous point-to-point messages on the local switched network. If we consider a perfect butterfly network we can minimize the number of synchronized steps that are needed to add the contribution to the scalar flux (Grama et al (2003)).

4.2 Optimizing computations with Adelie/SSI toolkit

In order to minimize the computational time, we used an optimized Linux kernel. This optimization is performed using Adelie/SSI (Adelie (2004)), a clustering toolkit over a Gentoo GNU/Linux base distribution (Gentoo (2004)). This allows the maximum level of performance to be reached without having to sacrifice the ease of system administration.

The Gentoo GNU/Linux distribution incorporates the concept of FreeBSD ports collection, named portage tree under Gentoo. This feature allows the complete system to be recompiled seamlessly without effort, permitting the highest level of optimization for a specific processor to be used. Gentoo also do away with the traditional System V init process in favor of a much more flexible named levels system. This system permits an almost infinite number of run-levels to be created and used.

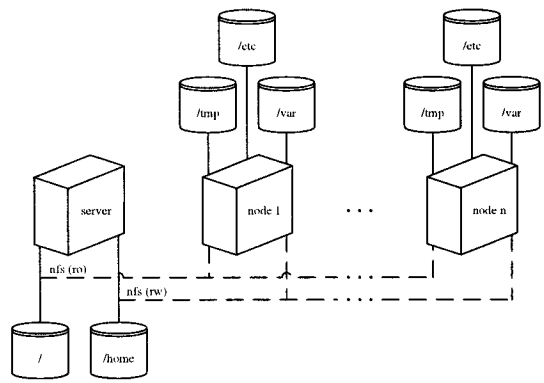

Figure 1: Adelie/SSI based cluster

Adelie/SSI, or Adelie for Single System Image, is an initiative of the Adelie Linux Project. This enhancement sits on top of a Gentoo system and allows a complete cluster to be run from a single copy of the system image. This image resides on the server disks and is shared between both the server and the disk-less nodes. Administration is done on this single image, while different run-levels allow customizing of the boot and init process for individual or groups of nodes. Since the nodes do not have disks nor

uses space on server disks for writing data, the integrity of a node can not be compromise from the node itself.

Adelie/SSI based cluster is composed of a server containing some form of storage system and a number of disk-less nodes (Fig. 1). The operating system image resides on the server and is exported to the nodes via a network file system. In order to have multiples nodes running from the same system image, some area of this image must be local to each node. These areas (`/etc`, `/tmp` and `/var`) are cloned initially from the system image into a memory based file system. After some modifications, they are then locally binded to the main system image. In this manner, it is possible to have a read-only system image which no node can corrupt.

5 Numerical results

All the results presented in this section are obtained for CANDU-6 3D supercell problems. A supercell is composed by two horizontal bundles containing the Uranium oxide fuel, and one vertical absorber rod Fig. 2. The transport equation is usually solved by critical buckling search with first order leakage treatment $B1$. Homogenization and condensation processes are then made with the resulting multigroup fluxes. Nuclear properties are condensed to a limited number of energy groups; only 2 groups are generally sufficient for good on-power followup in CANDU-6 reactor cores, keeping the small up-scattering effect from the thermal to fast group. For standard tests, we use 3 discrete directions and 89 energy groups. In all test cases presented here, we use the STRD option for the load balancing of the tracking file.

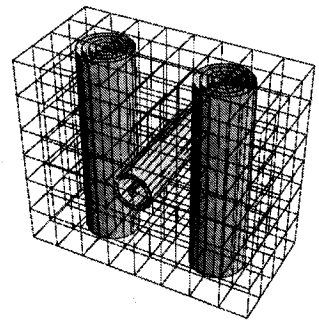

Figure 2: CANDU-6 supercell

Incr. cross-section	Relative errors (%)			
Number of processors	2	4	12	16
$\Delta\Sigma_t^1$	0.	0.	-0.008	-0.004
$\Delta\Sigma_a^1$	-0.01	0.006	-0.005	0.02
$\Delta\Sigma_{s0}^{1\leftarrow 1}$	-0.0008	0.007	-0.0076	0.005
$\Delta\Sigma_{s0}^{1\leftarrow 2}$	-0.002	0.008	0.005	0.004
$\Delta\Sigma_t^2$	0.008	0.04	0.01	-0.008
$\Delta\Sigma_a^2$	0.005	0.0062	0.013	0.004
$\Delta\Sigma_{s0}^{2\leftarrow 1}$	-0.01	0.05	0.031	0.02
$\Delta\Sigma_{s0}^{2\leftarrow 2}$	0.09	-0.02	0.08	-0.009

Table 1: Validation of the parallel algorithm on several processors

5.1 Validation of parallel program

The parallel program was run on several processors (from 1 up to 16 processors) and the results were compared with those given by the sequential solver. The relative error for a calculated value, presented in Table I, is defined as follows

$$\frac{\Delta\Sigma_{par} - \Delta\Sigma_{seq}}{\Delta\Sigma_{seq}} \times 100\%. \tag{10}$$

Results show that the relative errors, between the results given by the sequential code and the parallel one, are very small. The little discrepancies between the results are due to floating point accuracy. These differences can be explained by the fact that, when performing their partial addition on each processor, the summation order (adding first the biggest values or the smallest ones) can slightly affect the results by neglecting some small contributions.

5.2 Speedup

To evaluate the performance of our parallel program, we use the speedup S and the efficiency $\mathcal{E}$ as global metrics. The speedup is defined as usual by the ratio of the serial execution time to the parallel execution time

$$S = \frac{T_s}{T_p}; \tag{11}$$

and the efficiency of the system

$$\mathcal{E} = \frac{S}{N_p}. \tag{12}$$

As stated above, the sequential program is highly optimized and generates a single tracking file. The parallel program generates local tracking files (in the `/tmp` directory. The number of characteristics lines is linearly decreasing with the number of processors.
For our tests, we use the following machines:

- a Beowulf cluster of 16 disk-less nodes equipped with AMD Athlon 1.4 GHz processor and 1 GB memory connected by a fast Ethernet switched network. The message passing library installed is the LAM MPI implementation;
- a SMP cluster composed of 8 QuadXeon multiprocessors, each of these 8 nodes has 4 processors sharing a 4 GB memory. Nodes are connected with a Myrinet switch. Two MPI implementations are available on this machine: LAM and MPICH;
- a SGI shared memory machine with 16 IP35 Processors (500 MHz) sharing 16 GB memory connected by a Gigabit Ethernet switched network. The MPI implementation used is SGI proprietary version of MPI.
- a NUMA cluster composed of 32 nodes, each node has two AMD Opteron processors (2 GHz) sharing 5 GB memory. Nodes are connected with Myrinet Fiber. The MPI implementation used is MPICH.

In Table II, we show the speedups obtained with different parallel machines. The differences in the speedups are mainly due to the communication performances of each machine and to the MPI implementation used. This shows also that our parallel code is portable on different combination of architecture/implementation. Note that both clusters use Adelie/SSI.

Environment			No. of proc.			
Arch.	OS	implem	2	4	8	16
Beowulf ($P = 16$)	Linux	LAM	1.95	3.73	6.58	12.11
SMP ($P = 32$)	Linux	MPICH	1.98	3.92	7.93	15.29
NUMA ($P = 16$)	IRIX	SGIMPI	1.98	3.87	7.68	13.85
NUMA ($P = 64$)	Linux	MPICH	1.97	3.76	7.54	13.22

Table 2: Speedups using different parallel systems

5.3 The influence of the network characteristics on the communication time

For the same machine (SMP cluster), we study the influence of the network bandwidth on the performance. In Fig. 5.3, the speedups for two different implementations using respectively Fast Ethernet ($100Mb/s$) and Myrinet (over $1Gb/s$) switched networks are reported. The speedup curves shows that the speedups have been improved by using the Myrinet switch, especially when the number of processors becomes greater than 16 where the network traffic increases.

In order to extract the influence of the network on our application and to see if our parallel solver is optimal, we have studied more closely network characteristics. Table 5.3 presents measured network performance for various low latency interconnect.

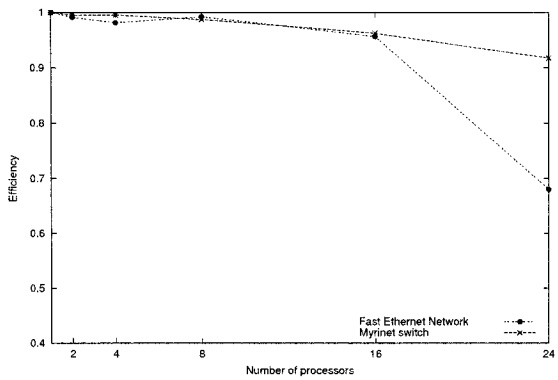

Figure 3: The efficiency for the Fast Ethernet and Myrinet switched networks

The numbers in bold were obtained using the MPI Performance Test version 1.3a benchmark (Gropp and Lusk (1999)) using two different clusters. The first is equiped with both Gigabit Ethernet and Myrinet Fiber, the second with Fast Ethernet and Myrinet Lan. These measures are completed with results obtained by different authors (Jiuxing Liu et al (2003); Seifert et al (2000); Baker et al (2001)). The results from Seifert et al (2000) were not used for VIA performances since the authors state that the latency measurements were unordinarly high due to the software implementation used.

$$T_{com} = \lceil \log N_p \rceil [t_s + N_g N_r t_w] \tag{13}$$

where N_p is the number of processors used for the computation, t_s is the message startup time for the data transfert also called the MPI latency, and t_w is the per-word transfert time and which is inversely proportional to the bandwidth.

For a fixed number of processors, the total communication time for our application is then a function of the network parameters

$$T_{com} = T_{com}(t_s, t_w) \tag{14}$$

In Fig. 5.3 and 5.3, we present the communication time versus the number of processors for those different networks presented in Table 5.3. In the first Figure, we present the results for a short message. The curves show that networks with a smallest per-word transfert time provide the best communication times. The best communication times here are better for the Infiniband, QsNet and the Myrinet(3) respectively. The same behavior is seen for a

longer message (Fig. 5.3). In general, $N_g N_r t_w >> t_s$; the communication time is then practically influenced by the per-word transfert time (or the bandwidth). The curves in these Figures could be classified according to the per-word transfert time corresponding to each network.
If we consider the machines with similar processors resources connected with those different networks, the best speedups expected will be those corresponding to the networks with the smallest per-word transfert time according to the famous Amdhal's law. Note that the choice of the machine connecting network should also be a compromise between the network bandwidth and the network cost.

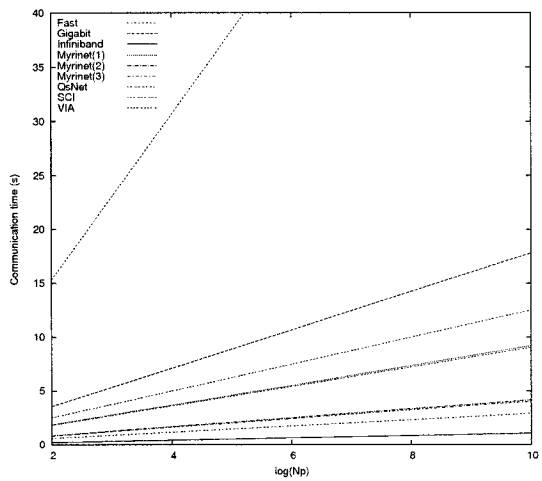

Figure 4: Communication time for different networks and with various number of processors (message size=0.46 MB)

6 Conclusion

The objective of this work was to evaluate the performance of the parallel program used to solve three-dimensional neutron transport problems. The message passing implementation using MPI, the particular setup using an optimized Linux kernel and the benefits of different load balancing techniques were presented. Speedups obtained yet are encouraging. It was also shown that the parallel code is portable on different types of parallel machines. This will allow us to implement our application on a small heterogeneous systems or using a computer grid.

We expect that our application will be scalable when we increase the dimensionality of the problem according with the increase of the number of processors. Therefore, we are currently working on improving the scalability potential of Adelie/SSI based systems. The goal is to

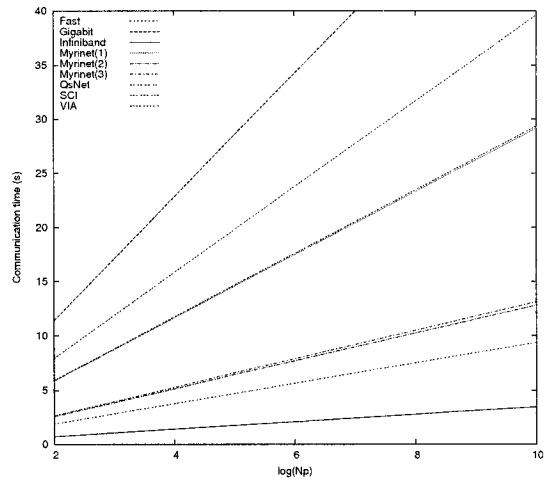

Figure 5: Communication time for different networks and with various number of processors (message size=1.28 MB)

support a massive ad hoc cluster easily, by federating a large number of workstations and clusters together. In order to acomplish this, we are developing a mecanism to rapidely and with no local modification convert a workstation into an Adelie/SSI node. Already existing Adelie/SSI servers and nodes will be integrated in the federated cluster resource tree. There is no need for a common architecture among nodes as heteregenous hardware will be supported.

The next step to performance analysis would be to analytically modelize the communication/calculation ratio in order to be able to predict the specific performance on different kind of architectures. In the coming years, it is expected that large-scale transport calculations will increase the accuracy of nuclear reactor simulations, with the direct effect of safer reactor design and operation. To perform these simulations, we will need scalable linear algorithms that are able to run on several thousands of nodes.

ACKNOWLEDGMENT

This work has been carried out partly with the help of a grant from the Natural Science and Engineering Research Council of Canada.

REFERENCES

Askew, J. R. (1972). 'A Characteristics Formulation of the Neutron Transport Equation in Complicated Geome-

Table 3: Comparison of different networks parameters for a fixed message size

Network	Theoretical bandwidth (Gb/s)	Measured latency (μs)	Measured transfer time per-word(ns)
Ethernet:			
Fast	**0.1**	**94.60**	**690.87**
Gigabit	**1.0**	**72.02**	**160.38**
Infiniband: *MT23108 HCA* Jiuxing et al(2003)	10.000	6.800	9.506
Myrinet:			
(1)M2L-PCI64A-2	**1.28**	**16.06**	**82.95**
(2)M3F-PCI64B-2	**2.0**	**13.39**	**37.56**
(3)M3F-(3)PCIXD-2 Jiuxing et al(2003)	2.0	6.7	36.45
QsNet: *Elan3* Jiuxing(2003)	2.72	4.6	26.04
SCI: *D310* Seifert et al(2000)	3.2	8.0	112.73
VIA : *cLAN1000* Baker et al(2001)	1.25	9.0	81.58

tries'. *Report AEEW-M 1108*, United Kingdom Atomic Energy Establishment,Winfrith.

Azmy, Y. Y. (1993). 'An Efficient Communication Scheme for Solving the S_N Equations on Message-Passing Multiprocessors'. *Trans. Am. Nucl. Soc.*, **69**, 203, 1993.

Baker, M., Farrell, P. A., Ong, H. and Scott, S. L. 'VIA Communication Performance on a Gigabit Ethernet Cluster', *Proceedings of the 7th International Euro-Par Conference on Parallel Processing*, Manchester, United Kingdom, August 2001.

Cyberlogic, (2004) 'Adelie Linux for Single System Image Cluster FAQ'. Available: http://www.adelielinux.com/en/ssi/faq.html.

Dahmani, M., Wu, G. J., Roy, R. and Koclas, J. (2002). 'Development and Parallelization of the Three-Dimensional Characteristics Solver MCI of DRAGON'. *Proc. of PHYSOR-2002*, Seoul, Korea, October 7-10.

Dahmani, M., Roy, R. and Koclas, J. (2003). 'Parallel Distribution of Tracking for 3D Neutron Transport Calculation'. *ANS Conf. on Nuclear Mathematical and Computational Sciences, on CD-ROM*, Gatlinburg, Tennessee, April 6-11, 2003.

Gentoo Technologies, Inc., 'Gentoo Linux – About Gentoo Linux'. [Online 2004]. Available: http://www.gentoo.org/main/en/about.xml.

Grama, A., Gupta, A., Karypis, G. and Kumar, V. (2003). 'Introduction to Parallel Computing'. *Second Edition, Addison-Wesley.*

Gropp, W. and Lusk, E. (1999). 'Reproducible measurements of MPI performance characteristics', *Technical Report ANL/MCS-P755-0699, Argonne National Labratory, Argonne*, IL, USA, June 1999.

Gropp, W., Lusk, E. and Skjellum, A. (1994). 'Using MPI: Portable Parallel Programming with the Message-Passing Interface', *MIT Press*, Cambridge.

Jiuxing Liu et al. (2003). 'Performance Comparison of MPI Implementations over Infiniband, Myrinet and Quadrics', *Proceedings of the ACM/IEEE SC2003 Conference*, Phoenix, Arizona, November 2003.

Kosaka, S. and Saji, E. (1999). 'The Characteristics Transport Calculation for a Multi-Assembly System Using Path Linking Technique'. *Proc. ANS Conference on Mathematics and Computation*, September 1999, Madrid, Spain.

Lee, G. S., Cho, N. Z. and Hong, S. G. (2000). 'Acceleration and Parallelization of the Method.of Characteristics for Lattice and Whole-Core Heterogeneous Calculation'. *Proc. of PHYSOR-2000 Conference*, 2000.

Qaddouri, A., Roy, R., Mayrand, M. and Goulard, B. (1996). 'Collision Probability Calculation and Multigroup Flux Solvers Using PVM'. *Nucl. Sc. Eng.*, **123**, 392, 1996.

Roy, R., Koclas, J., Shen, W., Jenkins, D. A., Altiparmakov, D. and Rouben, B. (2004) 'Reactor Core Simulations in Canada', *Proc. of PHYSOR-2004*, Chicago, April 25-27, 2004.

Roy, R. (2003) 'Théorie des probabilités de collision et méthode des caractéristiques', IGE-235, Institut de génie nucléaire, École Polytechnique de Montréal.

Seifert, F., Balkanski, D. and Rehm, W. (2000). 'Comparing MPI Performance of SCI and VIA', *Conference Proceeding of SCI-EUROPE 200*, Munich, Germany, August 2000.

Sjoden G. E. and Haghighat, A. (1997). 'PENTRAN-A 3-D Cartesian Parallel S_N Code with Angular, Energy, and Spatial Decomposition'. *Proc. Joint Int. Conf. Mathematical Methods and Supercomputing for Nuclear Applications*, Saratoga Springs, New York, October 5-6, 1997.